НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

«До захисту допущено»
Завідувач кафедри
_____ О.І. Ролік
«___»_____2019 р.

Дипломний проект
на здобуття ступеня бакалавра
з напряму підготовки 6. 050201 «Системна інженерія»
на тему: «Автоматизована система оцінювання знань учнів»

Виконав:
студент IV курсу, групи ІА-з51
Князєв Денис Сергійович _____

Керівник:
Ст. вик. Март Б.А. _____

Рецензент:
_____ _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2019 рік

АНОТАЦІЯ

Князєв Д.С. Автоматизована система оцінювання знань учнів. КПІ ім. Ігоря Сікорського, Київ, 2019.

Проект містить 67 с. тексту, 18 літературних джерел, рисунки, лістинг програмного коду та 4 конструкторських документа.

Ключові слова: оцінювання учнів, тестування, C#, ASP.NET Core, Entity Framework Core, база даних.

Об’єктом розробки є система оцінювання учнів для навчального закладу.

Мета розробки – покращення навчального процесу в закладах освіти.

Дипломна робота присвячена розробці автоматизованої системи оцінювання учнів для навчального закладу з використанням мови програмування C#. Результатом роботи став додаток, розроблений за допомогою технологій ASP.NET Core, Entity Framework Core, JavaScript, HTML та CSS. У роботі проводиться аналіз декількох основних систем аналогів, виявлено низка плюсів і мінусів, які були враховані при розробці програмного забезпечення. Описані можливі варіанти реалізації проекту загалом та окремих його компонентів. Було розглянуто загальну модель проекту та був повністю описаний функціонал системи та технології, обрані для розробки проекту.

В результаті була спроектована та побудована автоматизована система оцінювання учнів, яка може бути корисною для навчальних закладів.

SUMMARY

Kniaziev D. Automated student assessment system. Igor Sikorsky KPI, Kyiv, 2019.

The project contains 67 pages, 18 literary sources, drawings, code listing and 4 design documents.

Key words: student assessment, testing, C #, ASP.NET Core, Entity Framework Core, database.

The object of development is the system of assessment of students for an educational institution.

The purpose of the development is to improve the educational process in educational institutions.

This work is devoted to the development of an automated student assessment system for an educational institution using the C # programming language. The result was an application developed using ASP.NET Core, Entity Framework Core, JavaScript, HTML and CSS technologies. The work analyzes several basic similar systems, reveals several advantages and disadvantages that were taken into account when developing software. Described are possible options for project implementation in general and its individual components. The general model of the project was considered, and the functional system and technology chosen for project development were fully described.

As a result, an automated student assessment system was designed and built that could be useful for educational institutions.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту

на тему: «Автоматизована система оцінювання знань учнів»

Київ – 2019 рік

ЗМІСТ

ВСТУП

Контроль знань учнів шкіл, студентів вищих навчальних закладів та працівників відіграє у наш час велику роль. Контроль знань – це система науково та технічно обґрунтованої перевірки результатів навчання.

Існує багато видів контролю: контрольна, залік, іспит, атестація. Однак найбільш точним засобом виміру знань у наш час є тестування. Тест – це сукупність завдань, які спеціальним чином підготовлені та підібрані для того, щоб виявити рівень знань з певної теми. Одною з переваг тестів є те, що вони дозволяють протестувати всіх учасників в однакових умовах, застосовуючи при цьому одну і ту ж, заздалегідь розроблену шкалу оцінок, що значно підвищує об'єктивність і обґрунтованість оцінки в порівнянні, скажімо, з усним заліком.

Процес тестування отримав своє поширення в сферах, де необхідно точно визначити рівень знаннь по курсу навчання та де потрібен жорсткий відбір учасників: атестація і контроль знань учнів та студентів, атестація працівників підприємств, іспити з ПДР, екзаменаційний тест і.т.д.

На сьогоднішній день, в епоху глобальної комп'ютеризації, інформаційного світу, розвитку Інтернет і передачі даних все більш поширеними стають комп'ютерні системи тестування, які здатні замінити або доповнити старі методики викладання і методи контролю. У всіх сферах, де застосовувалися і застосовуються звичайні тести завдяки комп'ютерним системам проводити тестування стало набагато зручніше. Наприклад, дистанційна освіта, яка є популярною формою отримання знань.

Комп'ютерне тестування має багато переваг над традиційним тестуванням. Воно відрізняється високою продуктивністю та оперативністю процесу тестування і точністю результатів контролю знань – викладач має можливість проводити тестування набагато більшого числа учнів за менший час у порівнянні з усним опитуванням та може проаналізувати результати більш точно. Також перевагами є можливість самоконтролю студентів, адаптація

змісту і складності тестових запитань відповідно рівню знань, можливість застосувати в тестах мультимедійні завдання, підвищити відкритість процесу тестування, знизити фінансові та часові витрати на проведення тестування.

Однак на ряду з достоїнствами, у комп'ютерних тестів є і свої недоліки: знижується увага на оформлення рішення, втрачається інформація про процес виконання окремих завдань учнями, втрачається логіка міркування, підвищується ймовірність випадкового вибору відповіді, ставлення людей до комп'ютера не як до засобу отримання і контролю знань, а як до засобу розваги.

Робота включає п’ять розділів. Перший розділ описує предметну область питання оцінювання знань учнів. У другому розділі проводиться огляд існуючих рішень. Третій розділ включає опис основних технологій, що використовувались при розробці системи. Четвертий розділ описує модель даних системи. У п’ятому розділі проводиться опис реалізації системи та показ прикладу роботи системи.

СПИСОК ВИКОРИСТАНИХ У РОБОТІ СКОРОЧЕНЬ

LMS – Learning Management Systems;

VLE – Virtual Learning Environments;

LINQ – Language-Integrated Query;

MVC – Model-View-Controller;

HTTP – HyperText Transfer Protocol;

HTML – HyperText Markup Language;

ORM – Object-Relational Mapping;

IIS – Internet Information Server;

EDM – Entity Data Model;

IoC – Inversion of Control;

IDE – Integrated Development Environment;

DTD – Document Type Definition;

CSS – Cascading Style Sheets;

AJAX – Asynchronous Javascript and XML;

DNX – .NET Execution Environment;

SDK – Software Development Kit;

ПДР – Правила дорожнього руху.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Під час аналізу предметної області були виявлені наступні особливості системи тестування, які треба врахувати при розробці програмного продукту:

– Створення зручного та зрозумілого механізму реєстрації, аутентифікації та дизайну особистого кабінету користувача з підтримкою стандартів GDPR, а також двофакторної аутентифікації.

– Реалізація можливості створення та керування групами та учнями в них.

– Реалізація розбиття тестів на категорії за предметами.

– Реалізація зручного процесу тестування.

– Реалізація можливості пропуску та навігації поміж питаннями у тестах. Дуже часто учні пропускають складні питання і в першу чергу відповідають на прості.

– Реалізація різних налаштувань рандомізації для різних тестів. Наприклад, користувач має можливість встановити налаштування, яке визначає чи потрібно перемішувати порядок питань під час проходження тесту або налаштування, яке визначає чи потрібно перемішувати порядок варіантів відповідей у питаннях під час проходження тесту.

– Реалізація автоматичної перевірки правильності відповідей на тести.

– Використання реляційної бази даних для централізованого зберігання інформації (списків груп, тестових завдань, користувачів системи, результати тестування, тощо).

Виявлено п'ять основних форм тестових запитань, які є складовими для створення тестів з будь-якої навчальної дисципліни:

1) Запитання закритої форми.

Необхідно зазначити хрестиком або галочкою, обвести кружком потрібний варіант відповіді. Видів запитань закритої форми існує велика

кількість, але в основі лежить один принцип: студенту необхідно обрати відповідь на запитання з декількох запропонованих відповідей.

Наприклад:

3 + 3 дорівнює:

a) 6

b) 7

c) 8

2) Запитання відкритої форми.

На відміну від запитань закритої форми тут немає варіантів відповіді, а студенту необхідно дати чітку відповідь.

Наприклад:

3 + 3 = ?

3) Запитання на відповідність.

У запитаннях цього типу необхідно встановити відповідність між запропонованими варіантами в лівому і правому стовпчиках. Іноді справа кількість варіантів більше, ніж зліва, тобто передбачається, що деякі з них є в неправильними.

Наприклад:

2 + 3 – 8

4 - 2 – 5

6 + 2 – 7

____ – 2

4) Запитання на встановлення правильної послідовності.

Форма запитання, для відповіді на яку необхідно встановити послідовність будь-яких дій, подій, термінів і т.п.

Наприклад:

Встановити за зростанням вирази

5 - 2, 3 + 1, 2 * 3, 9 - 3

5) Запитання відкритої форми по принципу есе.

У запитаннях цього типу необхідно дати розгорнуту відповідь на поставлене запитання. На відміну від завдань відкритої форми, правильність відповіді перевіряється не системою, а викладачем.

Наприклад: Описати теорему Піфагора.

Можна також відзначити позитивні сторони тестування на конкретному прикладі. Наприклад, якщо під час оцінки результатів за кожне правильно виконане завдання ставиться один бал, то рівень знань з теми даного тесту буде виражений у певній кількості балів. Якщо розставити студентів, що здали тест за зростанням, ми отримаємо структурований ряд рівня знань з даної дисципліни. Якщо тестування проводиться регулярно протягом всього вивчення предмета, то в кінці викладач матиме чітке уявлення про рівень знань студентів даного курсу. Створюється рейтинг студентів з досліджуваної дисципліни.

2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Для огляду та аналізу існуючих рішень виберемо найбільш популярні та автоматизовані системи тестування, такі як PikaTest, UniTest, Indigo та Moodle.

Огляд системи тестування PikaTest

PikaTest – це безкоштовна програма для створення і проведення тестів з необмеженою кількістю питань. Вона є offline системою тестування.

Тест може містити мультимедійні файли, такі як аудіо-відео-файли, а також таблиці і зображення. Існує можливість додавання питання з варіантами відповідей і без них, а також визначення вартості кожного окремого питання. Можна створювати тести з обмеженим часом проходження. Файли тестів зберігаються в форматі *.pikatest. Докладний звіт про тестування зберігається в форматі *.txt.

Проходження тестування здійснюється за наступною схемою:

1) Реєстрація користувача;
2) Налаштування проходження тестування;
3) Проходження тестування;
4) Виведення докладної статистики про результати тестування з деталізацією до кожного питання;
5) Збереження звіту про результати тестування;
6) Відправка звіту по електронній пошті.

Основні переваги:

- система незалежна і має невеликий розмір;
- система є безкоштовною;
- програма легка для користування, і не вимагає якихось спеціальних знань;
- сумісна з усіма поширеними ОС;

Недоліки:

– система працює в режимі offline, при цьому довідка за програмою доступна тільки в інтернеті;
– відсутність чіткого поділу адміністратор-користувач;
– незахищеність даних;
– відсутність більшості корисних функцій.

Огляд системи тестування UniTest

UniTest – це комплексне програмне забезпечення для проведення тестування, функціонально реалізоване по CAM-технології (CAM – з англ. Computer-Aided Manufacturing).

Програма призначена для створення тестових завдань та організації процесу перевірки знань. Вона дозволяє організувати процес контролю знань за допомогою комп'ютерного тестування в мережі з використанням транспортного протоколу TCP / IP.

Система UniTest реалізована із застосуванням технологій .NET, унікальних алгоритмів паралельної обробки інформації та засобів криптографічного захисту.

Система UniTest складається з двох програмних модулів: UniTest TUTOR і UniTest CLIENT.

Модуль TUTOR підтримує роботу двох категорій користувачів («Адміністратор», «Тютор») і призначений для забезпечення роботи з тестовими завданнями, які крім текстової частини можуть включати в себе різні мультимедіа-файли: графіку, Flash-анімацію, відео і аудіо файли. Модуль TUTOR дозволяє організувати зберігання тестових завдань покатегоріям, а також створювати сценарії тестування, дає можливість ведення електронних журналів з результатами тестування, забезпечує розподіл учнів по підрозділах і групам.

Модуль CLIENT призначений для категорії користувачів «Учень», забезпечує доступ до віртуальної залікової книжки з результатами тестування та готових тестів системи UniTest.

Система є безкоштовною і доступна всім користувачам на офіційному сайті продукту.

Основні переваги:

- робота з програмою зручна і зрозуміла;
- зручний інтерфейс
- тести займають дуже мало місця;
- підтримка всіх основних і додаткових типів питань;
- високий рівень захисту даних;
- можливість проведення тестування як локально, так і по мережі;
- підтримка великої кількості мов;

Недоліки:

- несумісна з останніми версіями ОС;
- високі вимоги до технічних засобів;
- призначення логіна і пароля, для аутентифікації викладачів і студентів, безпосередньо «Адміністратором».

Огляд системи тестування Indigo

Indigo – являє собою комплекс програмного забезпечення, що дозволяє автоматизувати процес проведення тестування і обробки результатів. Система була розроблена в 2010 році.

Система «INDIGO» є універсальним інструментом, який можна використовувати для вирішення великої кількості завдань:

- визначення рівня готовності учнів шкіл до ДПА та ЗНО.
- тестування та контроль знань учнів з різних дисциплін.
- визначення професійного рівня працівників.
- Автоматизація психологічних тестів.

– Проведення опитувань.

– Автоматизація проведення олімпіад і вікторин.

Робота з INDIGO ділиться на дві частини: інтерфейс адміністратора і інтерфейс користувача.

Інтерфейс Адміністратора тестової оболонки являє собою Windows-додаток, яке реалізує наступні функції:

– Створення та редагування тестів.

– Управління тестами.

– Управління користувачами.

– Управління правилами тестування.

– Доступ до результатів тестування.

Інтерфейс користувача тестової оболонки являє собою Web-інтерфейс, які реалізує наступні функції:

– Реєстрація та авторизація в системі.

– Перегляд доступних тестів.

– Вибір тесту і проведення тестування.

– Перегляд результатів тестування.

– Доступ до журналу результатів.

Основні переваги:

– проста установка системи;

– доступний інтерфейс користувача;

– продукт сумісний з усіма ОС сімейства Windows;

– підтримка всіх поширених браузерів;

– централізоване зберігання даних;

– ієрархічна угруповання тестів і користувачів (правила тестування); широкі можливості конструктора тестів.

Недоліки:

– система є платною;

– відсутність поділу адміністратор-викладач (не завжди викладач має навички роботи з подібного роду системами);
– відносно великий обсяг споживаної пам'яті;
– високі вимоги до обладнання.

Огляд системи тестування Moodle

Moodle – це система управління вмістом сайту (Content Management System CMS), спеціально розроблена для створення онлайн-курсів викладачами. Такі системи часто називаються системами управління навчанням (Learning Management Systems – LMS) або віртуальними освітніми середовищами (Virtual Learning Environments – VLE). Moodle написана на мові програмування PHP і переведена на кілька десятків мов і використовується для навчання більш ніж в ста п'ятдесяти країнах світу.

Система тестування є лише частиною великої програми.

Тестування запропоновано здійснювати за такими етапами:

– Викладач розробляє і розміщує на сторінці свого курсу тести, вказуючи в їх параметрах дати, коли тести будуть доступними для проходження, час, який відводиться на виконання однієї спроби, кількість спроб, що надається кожному студенту і метод оцінювання.
– Викладач повідомляє студентам про зміст тесту, місце, дату та час тестування.
– Після тестування викладач аналізує його результат.

Moodle – це абсолютно безкоштовний проект з відкритим вихідним кодом. Його підтримкою займається компанія-розробник, штаб-квартира якої знаходиться в Австралії.

Основні переваги:

– повний набір необхідних функцій;
– відкритий вихідний код продукту (що дозволяє додати всі необхідні елементи);

– система Moodle універсальна в плані вимог (будь-яка ОС, встановлений модуль PHP і одна з СУБД);
– всі види тестів (включаючи написання есе).

Недоліки:

– система тестування є частиною великого програмного продукту;
– обслуговування надається за окрему плату.

3 ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ

Система розроблена з використанням мови програмування C# та найновіших версій технологій стеку серверної веб-розробки Microsoft: ASP.Net Core, Entity Framework Core. Розробка клієнтської частини включає використання мови програмування JavaScript та однієї з її найвідоміших бібліотек JQuery, мови розмітки гіпертексту HTML, та каскадних таблиць стилей CSS.

3.1 Основна інформація про мову програмування C#

Мова програмування C# була створена відносно недавно – в кінці 1998 року командою спеціалістів Microsoft.

ЇЇ ціллю була можливість створення різноманітних програм для платформ Microsoft.NET. Така прив’язаність платформ Microsoft .NET та мови C# в подальшому була усунена, проте сама платформа .NET у всіх версіях містить компілятор кодів даної мови програмування. Це дозволяє запускати додатки без налаштування додаткового програмного забезпечення.

Сама назва «C#» пов’язана з тим, що її синтаксис дуже нагадує C++. Загалом, C# перейняв багато позитивних рис своїх попередників - Delphi, C++, Java та інших. При цьому із C# були вилучені проблемні алгоритми.

Як відзначав творець мови, Андреас Хейлсберг, C# створювалася як мова компонентного програмування, і в цьому одне з головних переваг мови, спрямоване на можливість повторного використання створених компонентів. З інших об'єктивних факторів відзначимо наступні:

– C# повною мірою враховує всі можливості платформи .Net Framework

– C# – об'єктно-орієнтована мова, де типи, вбудовані в мову, представлені класами;
– C# – спадкоємець мов C та C++, який зберігає кращі риси цих мов програмування;
– завдяки каркасу Framework .Net, програмісти C# одержали порівняно тотожні переваги для роботи з віртуальною машиною, що й програмісти Java. Однак ефективність коду є вищою, оскільки виконавче середовище CLR являє собою компілятор проміжної мови, а віртуальна Java-машина є інтерпретатором байта-коду;
– бібліотека каркасів підтримує зручність побудови різних типів додатків на C#, дозволяючи легко будувати Web-додатки, інші види компонентів, досить просто зберігати й одержувати інформацію з бази даних й інших сховищ даних;
– реалізація, що сполучає побудову надійного й ефективного коду, є немаловажним чинником, що сприяє успіху C#.

C# продовжує активно розвиватись. З кожною новою версією з’являється все більше цікавих можливостей, як, наприклад, асинхронні методи, лямбда вирази, динамічне зв’язування тощо.

3.2 Основна інформація про платформу .NET Core

.NET Core – це модульна версія .NET Framework з можливістю переносу на інші платформи. Вона є підмножиною повної версії .NET Framework, та надає ключові можливості для реалізації функцій додатків, необхідних для повторного використання цього коду незалежно від цільової платформи.

.NET Core обладає наступними характеристиками:

– Кросплатформеність. Підтримка операційних систем Windows, macOS і Linux.

- Узгодженість між архітектурою. Однакове виконання коду в різних архітектурах, включаючи x64, x86 і ARM.
- Програми командного рядка. Зручні інструменти для локальної розробки і сценаріїв безперервної інтеграції.
- Гнучка розробка. Може включатися в додаток або встановлюватися паралельно (на рівні користувача або системи). Можливість використання з контейнерами Docker.
- Сумісність. Платформа .NET Core сумісна з .NET Framework, Xamarin і Mono завдяки .NET Standard.
- Відкритий код. Платформа .NET Core має відкритий код і поширюється за ліцензіями MIT і Apache 2. .NET Core є проектом .NET Foundation.
- Підтримка від Майкрософт. Корпорація Майкрософт надає підтримку .NET Core.

Продукт складається з декількох компонентів, кожен з яких може бути адаптований до нових платформ окремо і в різний час. Середовище виконання і основні бібліотеки, пов'язані з платформою, повинні переноситися як єдине ціле. Які не залежать від платформи бібліотеки повинні працювати "як є" на будь-якій платформі. В рамках проекту є тенденція до обмеження реалізацій, призначених для конкретних платформ, з метою підвищення ефективності розробки: якщо алгоритм або інтерфейс API можна повністю або частково реалізувати за допомогою незалежного від платформи коду C #, перевага віддається саме такого варіанту.

Сам .NET Core компонується з бібліотек, що носять назву "CoreFX", і з власне невеликого середовища виконання "CoreCLR".

CoreFX складається з набору бібліотек, що представляють інструменти з управління, доступом до консолі, колекціями, діагностикою, системою введення - виведення, LINQ, JSON, XML тощо. Кожна з бібліотек має

мінімальне число залежностей від будь-яких інших бібліотек, що покращує можливості переносу та розгортання пакетів CoreFX.

Для побудови додатків на базі .NET Core використовується кросплатформенне середовище виконання DNX. DNX – це середовище виконання і SDK, які необхідна для побудови та запуску додатків .NET на платформах Windows, MacOS і Linux. При цьому DNX запускати не тільки веб-додатки, але і консольні, а також нативні мобільні додатки. Таким чином, можна розробляти проект на одні платформі, а запускати на іншій, з умовою що на ній розгорнута сумісна версія DNX.

3.3 Особливості платформи ASP.NET Core

ASP.NET Core є кроссплатформенним, високопродуктивним середовищем з відкритим вихідним кодом для створення сучасних додатків, підключених до Інтернету. ASP.NET Core дозволяє виконувати наступні завдання:

- Створювати веб-додатки та служби, додатки IoT і серверні частини для мобільних додатків.
- Використовувати вибрані засоби розробки в Windows, macOS і Linux.
- Виконувати розгортання в хмарі або локальному середовищі.
- Працювати в .NET Core або .NET Framework.

При розгортанні для веб-додатку можна використовувати традиційний IIS. Але також можна запускати веб-додаток, використовуючи кросплатформенний веб-сервер Kestrel.

ASP.NET Core надає наступні переваги:

- Єдине рішення для створення призначеного для користувача веб-інтерфейсу і веб-API.

- Високий рівень тестованості коду.
- Razor Pages робить створення кодів сценаріїв для сторінок простішим і ефективнішим.
- Можливість розробки і запуску на платформах ОС Windows, macOS і Linux.
- Відкритий вихідний код і орієнтація на співтовариство.
- Інтеграція сучасних клієнтських платформ і робочих процесів розробки.
- Хмарна система конфігурації на основі середовища.
- Вбудоване введення залежностей.
- Спрощений високопродуктивний модульний конвеєр HTTP-запитів.
- Інструментарій, що спрощує процес сучасної веб-розробки.

3.4 Шаблон проектування MVC

Принцип MVC у веб-програмуванні (Model - View - Controller, Модель - Подання (Вид) - Контролер) - одна з найбільш поширених концепцій на сьогоднішній день. Він дозволяє розділити реалізацію логіки докладання, зовнішній вигляд (графічний інтерфейс, GUI) і взаємодію з користувачем. Це призводить до більш структурованого коду, дозволяє працювати над проектом більш спеціалізованим людям, спрощує підтримку коду, робить його більш логічним і зрозумілим. Зміна в одному з компонентів мінімально впливає на інші. Можна до однієї моделі підключати різні види, різні контролери. Розглянемо докладніше компоненти.

- Model (Модель) - містить так звану "Бізнес-логіку" - обробку і верифікацію даних, звернення до баз даних, представляє внутрішній устрій системи. Модель не повинна безпосередньо взаємодіяти з користувачем.

– View (Вид, Подання) описує зовнішній вигляд програми.
– Controller (Контролер) - сполучна ланка між моделлю і видом, отримує дані від користувача, передає їх моделі, отримує оброблений результат і передає його в уявлення

Взаємодію даних компонентів можна описати схемою, зображеною на рисунку 3.4.

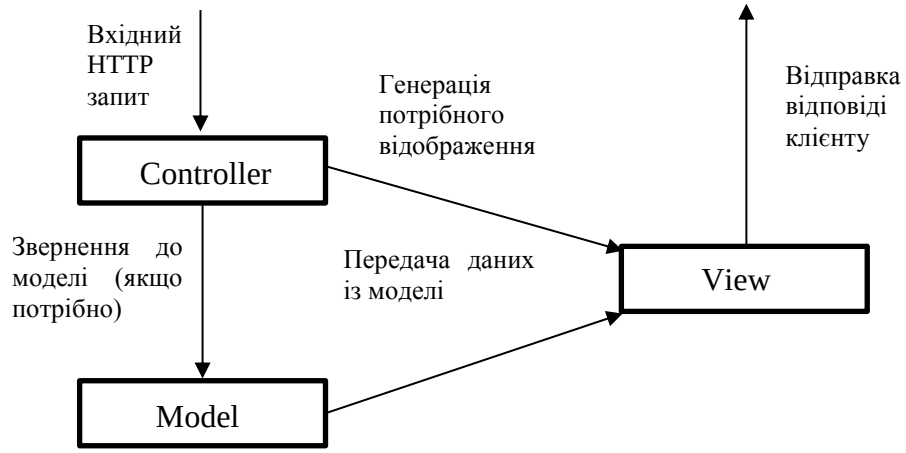

Рисунок 3.4 – Схема взаємодії компонентів MVC

Таке розмежування компонентів веб-додатку дозволяє реалізувати концепцію розділення відповідальності, при якій кожен компонент відповідає за свою строго відділену область функціональності системи. У зв'язку з чим легше побудувати роботу над окремими компонентами. І завдяки цьому додаток легше розробляти, підтримувати.

3.5 Особливості Entity Framework Core

Entity Framework (EF) Core - це проста, кроссплатформенна і розширена версія популярної технології доступу до даних Entity Framework з відкритим вихідним кодом. Вона може використовуватися як об'єктно реляційний модуль зіставлення (ORM), дозволяючи розробникам .NET працювати з базою даних за

допомогою об'єктів .NET і усуваючи необхідність в написанні здебільшого коду, необхідного для доступу до даних.

Entity Framework Core підтримує безліч різних систем баз даних. Таким чином, ми можемо через EF Core працювати з будь-якої СУБД, якщо для неї є потрібний провайдер.

За замовчуванням на даний момент Microsoft надає ряд вбудованих провайдерів: для роботи з MS SQL Server, для SQLite, для PostgreSQL. Також є провайдери від сторонніх постачальників, наприклад, для MySQL.

Також варто відзначити, що EF Core надає універсальний API для роботи з даними. І якщо, наприклад, ми вирішимо змінити цільову СУБД, то основні зміни в проекті будуть стосуватися насамперед конфігурації і настройки підключення до відповідних провайдерам. А код, який безпосередньо працює з даними, отримує дані, додає їх в БД і т.д., залишиться колишнім.

Центральної концепцією Entity Framework є поняття сутності або entity. Сутність визначає набір даних, які пов'язані з певним об'єктом. Тому дана технологія передбачає роботу не з таблицями, а з об'єктами і їх колекціями.

Відмінною рисою Entity Framework Core, як технології ORM, є використання запитів LINQ для вибірки даних з БД. За допомогою LINQ ми можемо створювати різні запити на вибірку об'єктів, в тому числі пов'язаних різними асоціативними зв'язками. А Entity Framework при виконання запиту транслює вираження LINQ в вирази, зрозумілі для конкретної СУБД (як правило, в вирази SQL).

Entity Framework передбачає три можливі способи взаємодії з базою даних:

– Database first: Entity Framework самостійно генерує набір класів, які відображають модель даних уже існуючої бази даних;

– Model first: розробник створює модель даних, на основі якої Entity Framework надалі генерує реальну базу даних на сервері;

– Code first: розробник створює класи моделі даних, які будуть зберігатися в базі даних. Далі Entity Framework на основі створених класів моделі генерує базу даних з відповідними таблицями.

3.6 Основна інформація про ASP.Net Identity

ASP.NET Identity представляє вбудовану в ASP.NET систему аутентифікації і авторизації. Дана система дозволяє користувачам створювати облікові записи, проводити автентифікацію, управляти обліковими записами або використовувати для входу на сайт облікові записи зовнішніх провайдерів, таких як Facebook, Google, Microsoft, Twitter та інших.. При створенні проекту в середовищі Visual Studio, розробнику дають можливість вибрати один із наступних типів автентифікації:

– No Authentication: ASP.Net Identity і вбудована система автентифікації відсутня;

– Individual User Accounts: проект за замовчуванням включає систему ASP.Net Identity, яка дозволяє авторизуватись користувачам як всередині програми, так і за допомогою зовнішніх сервісів;

– Organizational Accounts: підходить для сайтів та веб-додатків окремих компаній та організацій;

– Windows Authentication: система автентифікації за допомогою облікових записів Windows;

Після вибору відповідного типу автентифікації, автоматично у проект включаються всі необхідні залежності та створюється початковий шаблон.

3.7 Основна інформація про мову програмування JavaScript

JavaScript – прототипно-орієнтована мова програмування із С-подібним синтаксисом. Сьогоднішній світ веб-сайтів важко уявити без мови JavaScript.

JavaScript - це те, що робить живими веб-сторінки, які ми кожен день переглядаємо в своєму веб-браузері. Прототипне програмування – стиль об’єктно-орієнтованого програмування, де відсутнє поняття класу, а наслідування відбувається шляхом клонування існуючого примірника об’єкта – прототипу.

Основні архітектурні риси JavaScript: динамічна слабка типізація, автоматичне управління пам’яттю, прототипне програмування. Найбільш широко використовується у веб-розробці, як мова сценаріїв, для надання інтерактивності веб-сторінкам. Працюючи на стороні клієнта, JavaScript може керувати розподілом даних, дизайном, реакцією на певні події тощо.

З самого початку існувало кілька веб-браузерів (Netscape, Internet Explorer), які надавали різні реалізації мови. І щоб звести різні реалізації до загального стрижня і стандартизувати мову під керівництвом організації ECMA був розроблений стандарт ECMAScript. В принципі самі терміни JavaScript і ECMAScript є багато в чому взаємозамінними і відносяться до одного і того ж мови.

JavaScript є мовою, що інтерпретується. Це означає, що код на мові JavaScript виконується за допомогою інтерпретатора. Інтерпретатор отримує інструкції мови JavaScript, які визначені на веб-сторінці, виконує їх (або інтерпретує).

Мова JavaScript активно розвивається у наш час, як і веб-технології в цілому. А нові стандарти ECMAScript сприяють лише покращенню та розширенню функціональних можливостей та синтаксису мови.

3.8 Мова розмітки гіпертексту HTML

HTML (HyperText Markup Language) представляє мову розмітки гіпертексту, яка використовується переважно для створення документів в мережі інтернет. HTML почав свій шлях на початку 90-х років як примітивний

мову для створення веб-сторінок, і зараз вже важко уявити собі інтернет без HTML. Мову було створено на основі мови SGML, що пояснює концепцію визначення типу документу та структурної розмітки тексту. Поряд з CSS та JavaScript, HTML є фундаментом технологій створення веб-сторінок та користувацьких інтерфейсів для мобільних та веб-додатків.

Структурні елементи мови формуються за допомогою використання тегів (рисунок 3.8).

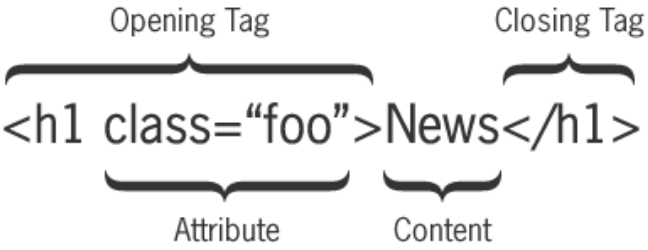

Рисунок 3.8 – Структура тегів HTML

Теги, такі як <img> та <input> визначають контент сторінки. Інші, такі як <p> та <h1> визначають семантичний розподіл тексту сторінки. Також одні теги можуть включати в себе інші, в якості допоміжних елементів. Браузери не відображають теги, але використовують їх для інтерпретації вмісту сторінки.

Як і природна мова, комп'ютерна мова має свої основні елементи: словник, синтаксис та граматика. HTML використовує машинно-зчитуючу граматику, яка називається DTD – механізм, успадкований від SGML. Процес перевірки документа на дотримання прави визначених мовою називається валідацією, а інструмент, що здійснює перевірку – валідатором. Документ, що не містить помилок, називають валідним. Згідно з цією концепцією, валідація HTML документу визначається, як процес його перевірки за правилами граматики визначеними в DTD, посилання на які визначені структурним елементом doctype.

У 2014 році офіційно була завершена робота над новим стандартом - HTML5, який фактично зробив революцію, несучи в HTML багато нового, а саме:

- визначає новий алгоритм парсинга для створення структури DOM;
- додавання нових елементів і тегів, як наприклад, елементи video, audio і ряд інших;
- перевизначення правил і семантики вже існуючих елементів HTML;

3.9 Каскадні таблиці стилей CSS

CSS – це потужний стандарт на основі текстового формату, який визначає уявлення даних в браузері. Якщо формат HTML надає інформацію про склад документа, то таблиці стилів повідомляють як він повинен виглядати.

Таким чином каскадні таблиці стилів дають можливість зберігати вміст окремо від його уявлення. Зазвичай використовується для оформлення зовнішнього виду веб сторінок, написаних за допомогою HTML.

Стиль включає всі типи елементів дизайну: шрифт, фон, текст, кольори посилань, поля і розташування об'єктів на сторінці. CSS розроблялися так, щоб забезпечити більший рівень контролю над розміщенням тексту і графіки.

Каскадні таблиці стилів забезпечують належний рівень єдності оформлення, організації і контролю під час розробки вузла, який є недосяжним за допомогою одного тільки HTML.

Правило CSS (рисунок 3.9) складається з двох основних частин: селектор, який вибирає ті структурні частини документу, до яких правило застосовується і блок декларації, де оголошують значення властивостей.

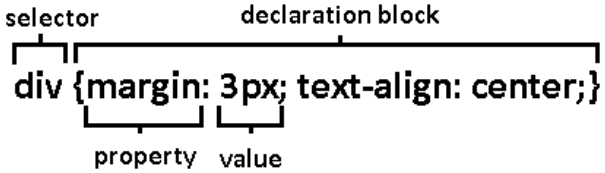

Рисунок 3.9 – Структура правила в CSS

Існує декілька видів селекторів, серед яких найбільш популярними є: універсальний селектор, селектор класів, селектор ідентифікаторів, селектор елементів, селектор нащадків та селектор дочірніх елементів.

Головна відмінність між класом та ідентифікатором в тому, що окремий клас може бути присвоєний декільком елементам в межах документу, а ідентифікатор – унікальне значення в межах документу, що присвоюється тільки одному елементу. Також відмінність у тому , що можуть існувати множинні класи, коли клас окремого елемента складається з декількох слів, розділених пробілами. У такому випадку до елемента будуть застосовані правила для усіх класів у множині.

3.10 Середовище розробки Visual Studio

В якості середовища розробки обрано Microsoft Visual Studio 2017, що надає весь необхідний функціонал для розробки веб-додатків на основі платформи .NET Core.

Microsoft Visual Studio – серія програмних продуктів компанії Microsoft для розробки програмного забезпечення. Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторінга коду.

Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і відладчик машинного рівня. Решта вбудовуються інструменти включають в себе редактор форм для спрощення створення графічного інтерфейсу додатку, веб-редактор, дизайнер класів і дизайнер схеми бази даних. Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (як, наприклад, Subversion і Visual SourceSafe), додавання нових наборів інструментів

(наприклад, для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування) або інструментів для інших аспектів процесу розробки програмного забезпечення (наприклад, клієнт Team Explorer для роботи з Team Foundation Server).

Присутня вбудована підтримка різних мов програмування, таких як C, C++, Visual Basic, F#, C#. Підтримку інших мов, таких як Python, Ruby, M можна забезпечити за допомогою підключення відповідних мовних служб, встановлених окремо.

4 МОДЕЛЬ ДАНИХ СИСТЕМИ

4.1 Опис моделі даних

Модель даних – це самодостатнє, абстрактне та логічне визначення певної групи об’єктів, які є доменом для створення бізнес-логіки програми.

4.1.1 Основні компоненти сутностей

При використанні підходу Code First, створений клас моделі даних може містити наступні компоненти:

– первинний ключ – унікальний ідентифікатор сутності, що однозначно визначає об’єкт. Для кожного об’єкта в межах колекції це завжди унікальне значення;

– зовнішній ключ –властивість, що визначає асоціативний зв’язок між сутностями. Значення первинного ключа певного об’єкта відповідає зовнішньому ключу сутності;

– навігаційна властивість – властивість, яка дозволяє отримати екземпляр об’єкту чи групу об’єктів, які пов’язані зв’язками з даною сутністю. Вона формується на основі зовнішніх ключів;

– атрибути – компоненти, що дозволяють додати метадані до певних компонентів сутностей у коді. Наприклад: атрибут [Length] вказує на те, яку максимальну або мінімальну довжину може мати поле сутності, атрибут [Required] вказує на те, що поле є обов’язковим.

Особливістю Entity Framework та підходу Code First є використання угод при створенні моделі:

– всі первинні ключі за замовчуванням є ненульовими;

– властивості, що містять значимі типи даних або тип Nullable<T> є ненульовими, всі інші – нульовими;

– властивість з ім’ям «Id» або «НазваСутностіId» за замовчуванням є первинним ключем. Наприклад: властивість QuestionId в класі Question, властивість Id в класі Student;

– властивість з ім’ям «НазваІншоїСутностіId» за замовчування є зовнішнім ключем;

– назви таблиць в базі даних, визначаються як назва класу у формі множини. Наприклад: клас Quetion – таблиця Questions, клас Test – таблиця Tests;

– назви стовбців таблиці в базі даних відповідають назвам властивостей класу моделі;

Угоди можна перевизначити з допомогою Fluent API або атрибутів, що дозволяє розробляти гнучкі та функціональні моделі даних.

4.1.2 Базовий клас Entity та інтерфейс ITrackable

Базовим класом для всіх сутностей домену додатку є клас Entity, який реалізує інтерфейс ITrackable. Інтерфейс ITrackable розроблений для аудиту змін у сутностях системи. Ці дані можуть використовуватись для налагодження системи, пошуку несправностей, статистики та побудови звітів.

Клас Entity (рисунок 4.1) міститить наступні компоненти:

– первинний ключ, для унікальної ідентифікації сутностей;

– дати створення та модифікації сутності (компоненти інтерфейсу ITrackable);

Всі сутності, що будуть приведені надалі містять ці компоненти, тому вони не будуть повторно описуватись при розгляді кожної окремої сутності.

Рисунок 4.1 – Клас Entity

4.1.3 Сутність «Тест»

Сутність «Тест» – ключова сутність системи. Адміністратор системи може створювати, змінювати та видаляти тести, а також відкривати доступ для проходження тестів окремим групам учнів.

Сутність «Тест» містить наступні компоненти:

– назва тесту;

– тривалість тесту;

– посилання на предмет до якого належить тест;

– логічне значення, яке визначає чи потрібно перемішувати порядок питань під час проходження тесту;

– логічне значення, яке визначає чи потрібно перемішувати порядок варіантів відповідей у питаннях під час проходження тесту;

– перелік питань, що входять до складу тесту;

– перелік груп учнів, яким надано доступ для проходження тесту;

– перелік тестових сесій, що були виконані учнями;

Для представлення сутності у програмному коді було створено клас Test (рисунок 4.2).

Рисунок 4.2 – Клас Test

4.1.4 Сутність «Питання»

Сутність «Питання» являє собою основу для формування контенту тесту. Адміністратор системи може створювати, змінювати та видаляти питання з тесту. Система підтримує п'ять типів питань: закритої форми, відкритої форми, питання на відповідність, на встановлення правильної послідовності, відкритої форми за принципом есе.

Сутність «Питання» містить наступні компоненти:

- тип питання;
- текст питання;
- посилання на тест до якого належить питання;

– дані питання, в залежності від типу питання існує визначений формат даних;

– відповідь на питання, в залежності від типу питання існує визначений формат відповіді;

Для представлення сутності у програмному коді було створено клас Question (рисунок 4.3).

Рисунок 4.3 – Клас Question

4.1.5 Сутність «Предмет»

Сутність «Предмет» являється допоміжною, та призначена для додаткової класифікації тестів за предметом . Адміністратор системи може створювати, змінювати та видаляти предмети.

Сутність «Предмет» містить наступні компоненти:

– назва предмету;

– перелік тестів з даного предмету;

Для представлення сутності у програмному коді було створено клас Subject (рисунок 4.4).

Рисунок 4.4 – Клас Subject

4.1.6 Сутність «Студент»

Сутність «Студент» являє собою відображення студента-користувача у системі. Адміністратор системи може лише змінювати та видаляти студентів, але не створювати. Створення студентів доступно тільки через реєстрацію.

Сутність «Студент» містить наступні компоненти:

– ім'я студента;

– посилання на групу до якого належить студент;

– перелік тестових сесій, що були виконані студентом;

– унікальний ідентифікатор користувача системи, який призначений для зв'язку з аккаунтом у системі.

Для представлення сутності у програмному коді було створено клас Student (рисунок 4.5).

Рисунок 4.5 – Клас Student

4.1.7 Сутність «Група»

Сутність «Група» являє собою відображення групи студентів у системі. Адміністратор системи може створювати, змінювати та видаляти групи.

Сутність «Група» містить наступні компоненти:

– назва групи;

– перелік студентів, що належать до групи;

– перелік тестів, доступ для проходження яких надано групі;

Для представлення сутності у програмному коді було створено клас Group (рисунок 4.6).

Рисунок 4.6 – Клас Group

4.1.8 Сутність «Тестова сесія»

Сутність «Тестова сесія» являє собою відображення тестової сесії у системі. Адміністратор системи може лише переглядати інформацію про тестові сесії. Створення, запуск за завершення сесії доступні тільки через інтерфейс студента. Сутність може перебувати у двох станах: «у процесі» та «завершена».

Сутність «Тестова сесія» містить наступні компоненти:

- стан тестової сесії;
- дата початку тестової сесії;
- дата завершення тестової сесії;
- час, який залишився для проходження тесту;
- посилання на студента до якого належить тестова сесія;
- посилання на тест;
- перелік відповідей на питання тесту;

Для представлення сутності у програмному коді було створено клас TestSession (рисунок 4.7).

Рисунок 4.7 – Клас TestSession

4.1.9 Сутність «Відповідь на питання»

Сутність «Відповідь на питання» являє собою відображення відповіді на питання тесту в системі. Адміністратор системи може лише переглядати інформацію про відповіді на питання, а також перевіряти їх правильність. Створення та корегування відповідей на питання доступні тільки через інтерфейс студента.

Сутність «Відповідь на питання» містить наступні компоненти:

– дані відповіді на питання, в залежності від типу питання існує визначений формат даних;

– логічне значення, яке визначає правильність відповіді;

– посилання на тестову сесію в рамках якої було дана відповідь;

– посилання на питання до якого була дана відповідь;

Для представлення сутності у програмному коді було створено клас QuestionAnswer (рисунок 4.8).

Рисунок 4.8 – Клас QuestionAnswer

4.2 Діаграма моделі даних

Після збереження та проведення виконання міграції схем та даних, на основі розроблених класів була створена база даних з відповідною моделлю даних (ІА51.060БАК.005 Д1 Схема бази даних).

5 ОПИС РЕАЛІЗАЦІЇ СИСТЕМИ

5.1 Реєстрація та аутентифікація користувачів

Робота з системою розпочинається з реєстрацій власного профілю користувача. У системі існує дві ролі: адміністратор та студент. Для реєстрації профілю адміністратора у вікні реєстрації (рисунок 5.1) користувач вводить свої дані, а саме адресу електронної пошти, пароль, його копію та токен адміністратора. Цей токен зберігається у конфігурації системи у зашифрованому вигляді. Приклад токену: ec06ba92-4017-e082-32bd-ad0756fed804.

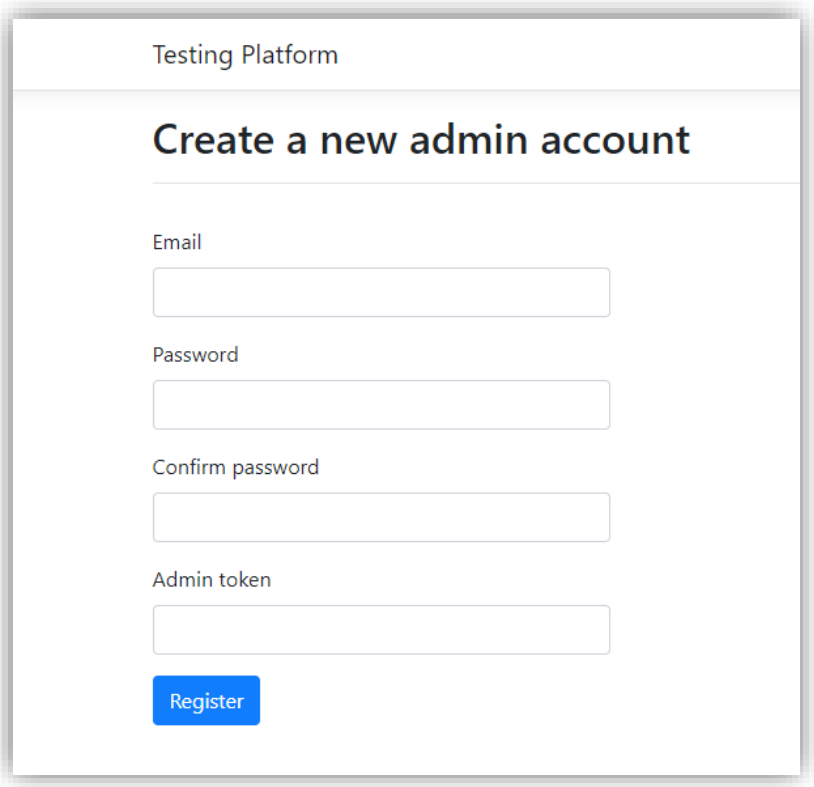

Рисунок 5.1 – Вікно реєстрації профілю адміністратора

Для реєстрації профілю студента у вікні реєстрації (рисунок 5.2) користувач вводить свої дані, а саме адресу електронної пошти, пароль, його

копію, ПІБ та обирає групу з списку запропонованих. Списки груп налаштовуються адміністратором системи.

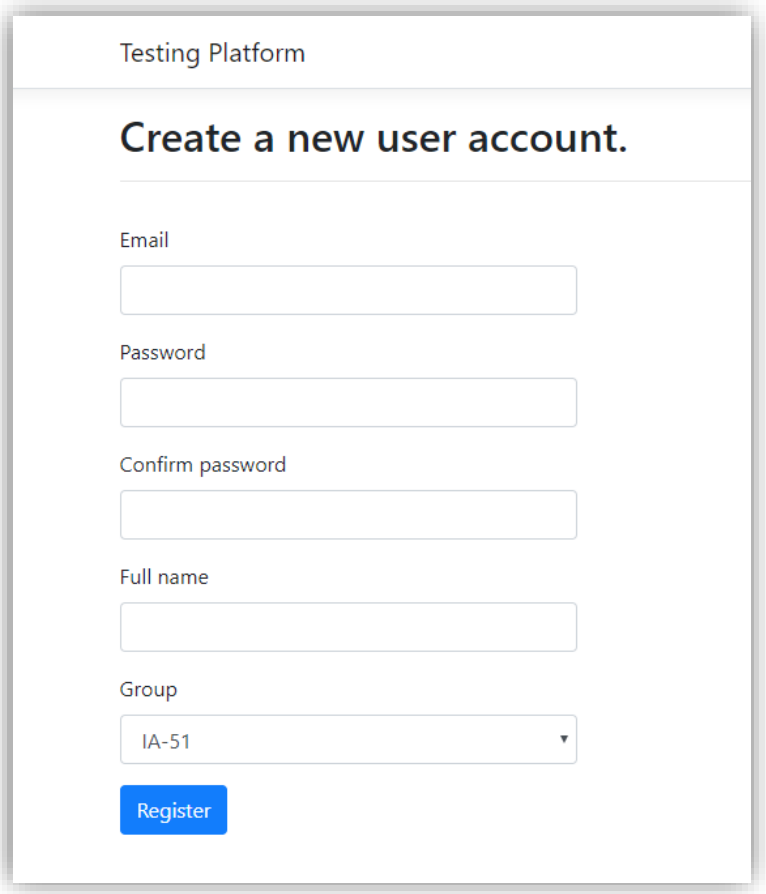

Рисунок 5.2 – Вікно реєстрації профілю студента

Після вдалої реєстрації на електронну пошту, яка була вказана під час створення профілю висилається лист для підтвердження її приналежності користувачу.

Для входу в систему користувач переходить до вікна аутентифікації (рисунок 5.3) та вводить свою адресу електронної пошти та пароль. Додатково користувач може обрати опцію «Запам’ятати мене» для автоматичної аутентифікації при наступну вході в систему. У випадку, якщо користувач забув свій пароль у системі доступна можливість відновити його обрав опцію «Забули свій пароль?» у вікні аутентифікації.

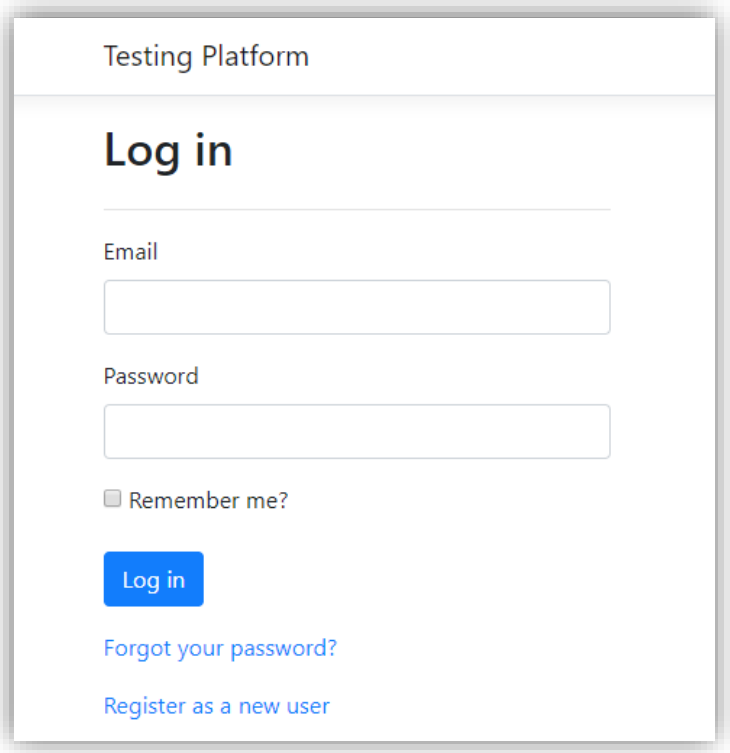

Рисунок 5.3 – Вікно аутентифікації

5.2 Особистий кабінет користувача

Після вдалої аутентифікації користувач має можливість перейти до особистого кабінету. У вікні налаштувань особистого кабінету реалізовані чотири категорії налаштувань:

– «Профіль»;
– «Пароль»;
– «Двофакторна аутентифікація»;
– «Персональні дані».

У вікні налаштувань «Профіль» (рисунок 5.4) користувач має можливість змінити адресу електронної пошти, повторно вислати лист для її підтвердження, а також зберегти номер мобільного телефону.

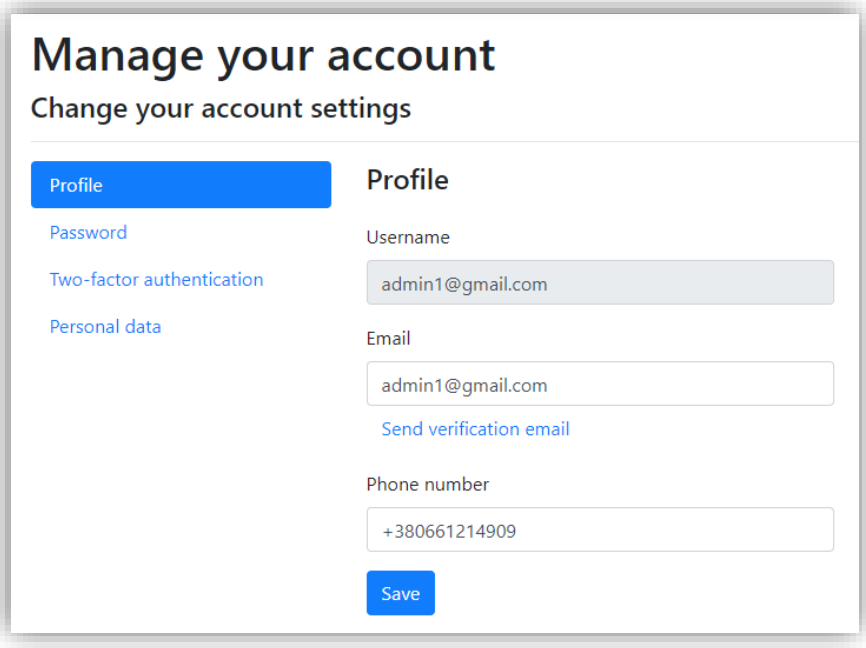

Рисунок 5.4 – Вікно налаштувань «Профіль»

Перейшовши до вікна налаштувань «Пароль» (рисунок 5.5) користувач має можливість змінити пароль. Для цього користувачу потрібно ввести попередній пароль, новий пароль та його копію.

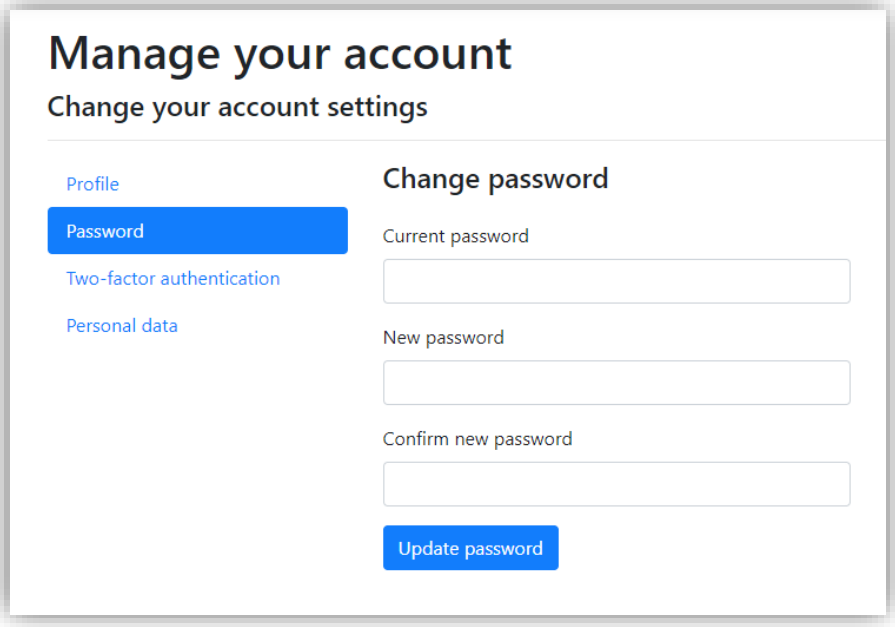

Рисунок 5.5 – Вікно налаштувань «Пароль»

Для включення двофакторної аутентифікації користувач переходить до вікна налаштувань «Двофакторна аутентифікація» (рисунок 5.6). Користувачу потрібно ввести код-ключ до спеціального додатку, такого як Microsoft Authenticator або Google Authenticator. Наступним етапом є копіювання тимчасового коду з додатку у спеціальне поле у вікні системи. Під час наступної аутентифікації потрібно буде додатково вказувати новий тимчасовий код з додатку. Після включення двофакторної аутентифікації система генерує спеціальні коди доступу для входу без використання додатку. Приклад коду: b064084d. Користувач у тому ж вікні має можливість відключити двофакторну аутентифікацію або перегенерувати коди доступу.

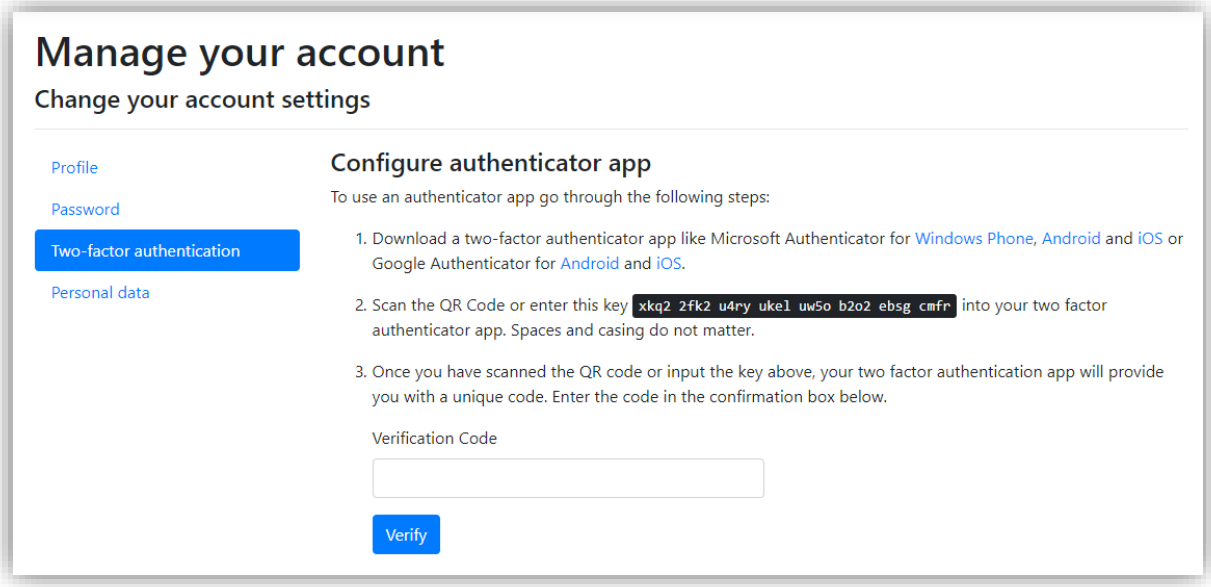

Рисунок 5.6 – Вікно налаштувань «Двофакторна аутентифікація»

Вікно налаштувань «Персональні дані» (рисунок 5.7) було розроблено для підтримки системою стандартів GDPR. Користувач має можливість переглянути всю інформацію про себе у системі, а також видалити свій обліковий запис з системи. Приклад звіту про персональні дані: {"Id": "e7295f4f-1776-4d47-9912-c4450a8c63ac", "UserName": "admin1@gmail.com", "Email": "admin1@gmail.com", "EmailConfirmed": "False", "PhoneNumber":

"+380661214909", "PhoneNumberConfirmed": "False", "TwoFactorEnabled": "True"}.

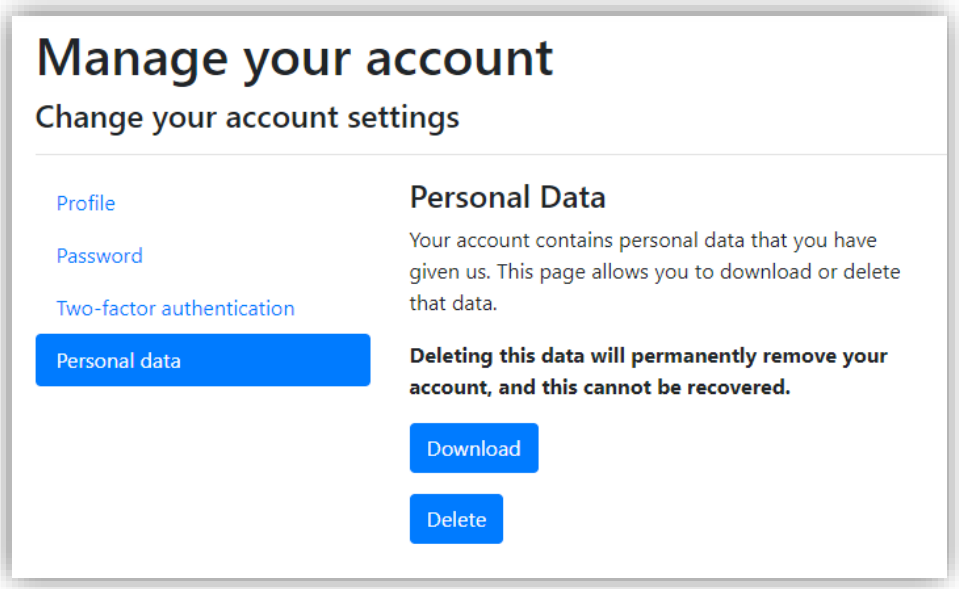

Рисунок 5.7 – Вікно налаштувань «Персональні дані»

5.3 Робочій простір адміністратора

Користувач системи з правами адміністратора має доступ до наступних розділів системи:

– «Студенти»;
– «Групи»;
– «Предмети»;
– «Тести»;
– «Тестові сесії».

Навігація між розділами системи проводиться через навігаційну панель (рисунок 5.8) у верхній частині сторінки.

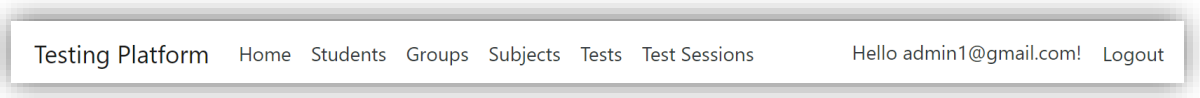

Рисунок 5.8 – Навігаційна панель адміністратора

5.3.1 Розділ «Студенти»

Перейшовши до розділу «Студенти» користувачу відкривається вікно зі списком студентів (рисунок 5.9). Для кожного студента відображається ПІБ, група та доступні дві опції: «Редагування», «Видалення». Створення студентів доступно тільки через реєстрацію. Перейшовши до вікна «Редагування» (рисунок 5.10) користувач має можливість змінити ПІБ та групу студента.

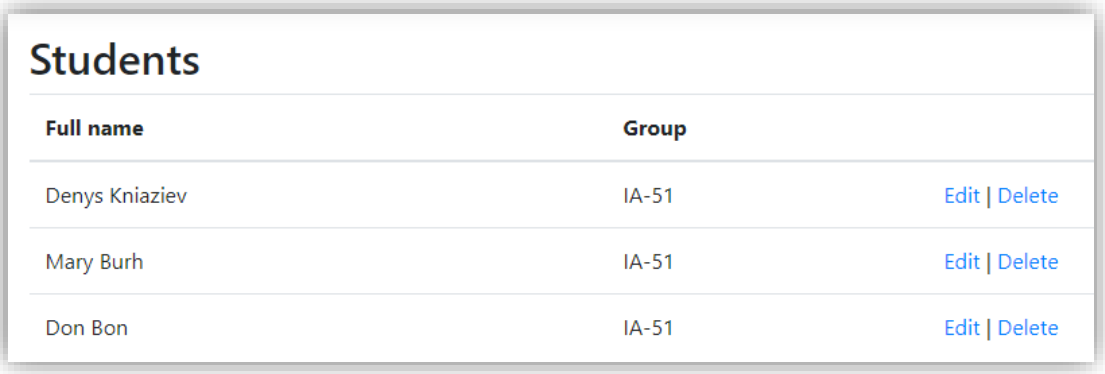

Рисунок 5.9 – Вікно зі списком студентів

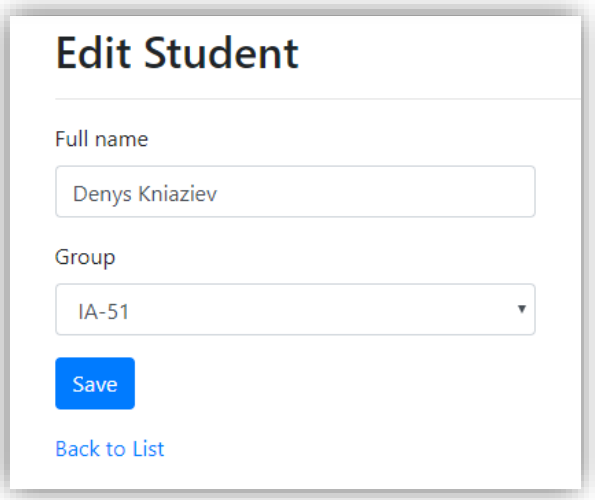

Рисунок 5.10 – Вікно редагування студента

5.3.2 Розділ «Групи»

Перейшовши до розділу «Групи» користувачу відкривається вікно зі списком груп (рисунок 5.11). Для кожної групи відображається назва та доступні три опції: «Детальніше», «Редагування», «Видалення». Також користувач може додати нову групу застосувавши опцію «Створити нову».

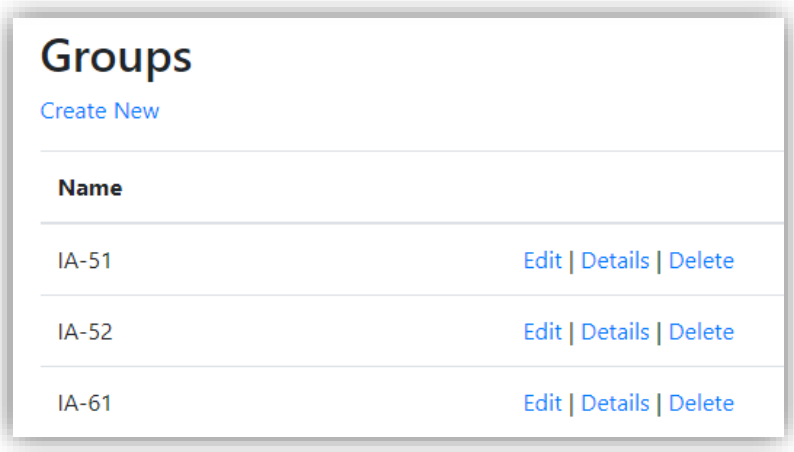

Рисунок 5.11 – Вікно зі списком груп

Обравши опцію «Створити нову» або «Редагування» користувач переходить до вікна створення (рисунок 5.12) та редагування групи відповідно. Вікна мають подібний вигляд. Користувач має можливість змінити назву групи.

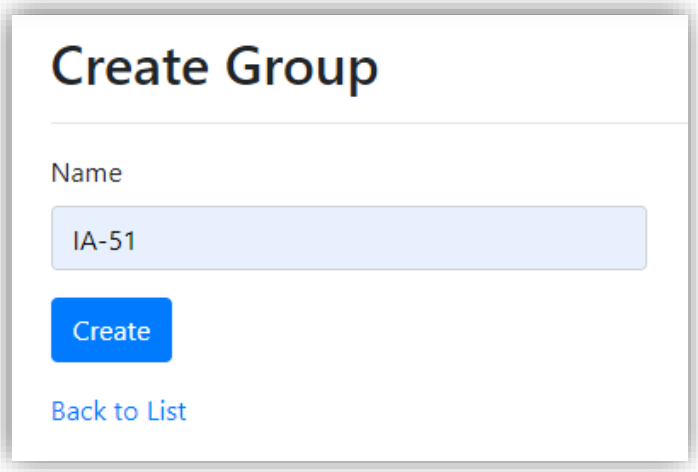

Рисунок 5.12 – Вікно створення групи

Обравши опцію «Детальніше» користувач переходить до вікна деталей групи (рисунок 5.13) та має можливість переглянути назву групи, список студентів, які належать до цієї групи, а також список тестів, що відкриті для проходження студентам групи.

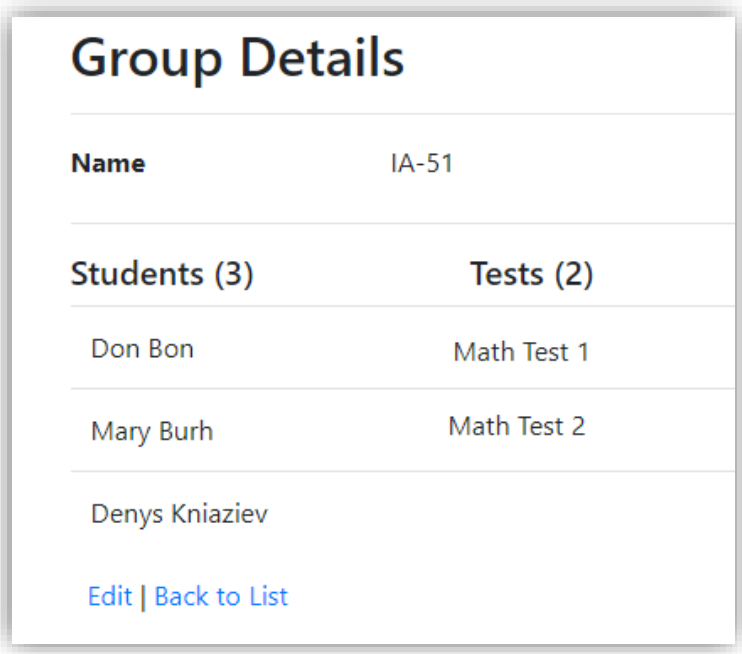

Рисунок 5.13 – Вікно деталей групи

5.3.3 Розділ «Предмети»

Перейшовши до розділу «Предмети» користувачу відкривається вікно зі списком предметів (рисунок 5.14). Для кожного предмету відображається назва та доступні три опції: «Детальніше», «Редагування», «Видалення». Також користувач може додати новий предмет застосувавши опцію «Створити новий».

Обравши опцію «Створити новий» або «Редагування» користувач переходить до вікна створення (рисунок 5.15) та редагування предмету відповідно. Вікна мають подібний вигляд. Користувач має можливість змінити назву предмету.

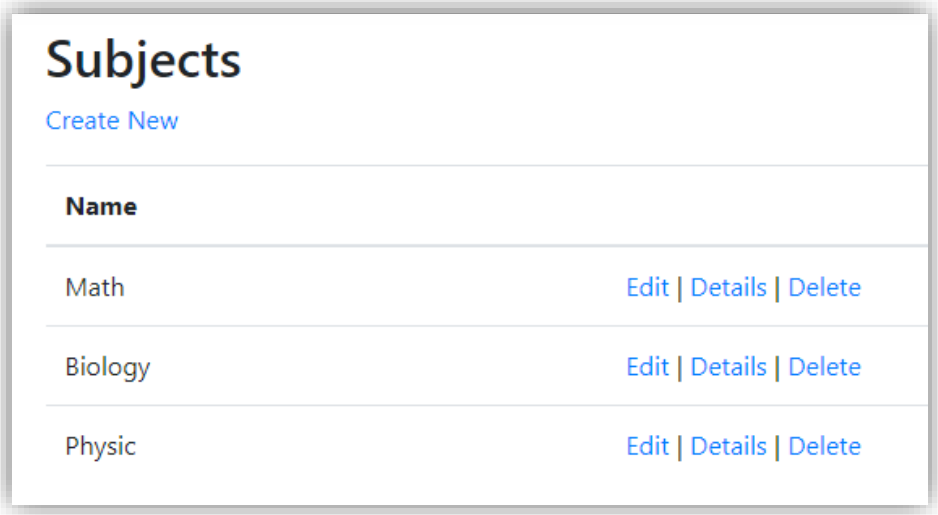

Рисунок 5.14 – Вікно зі списком предметів

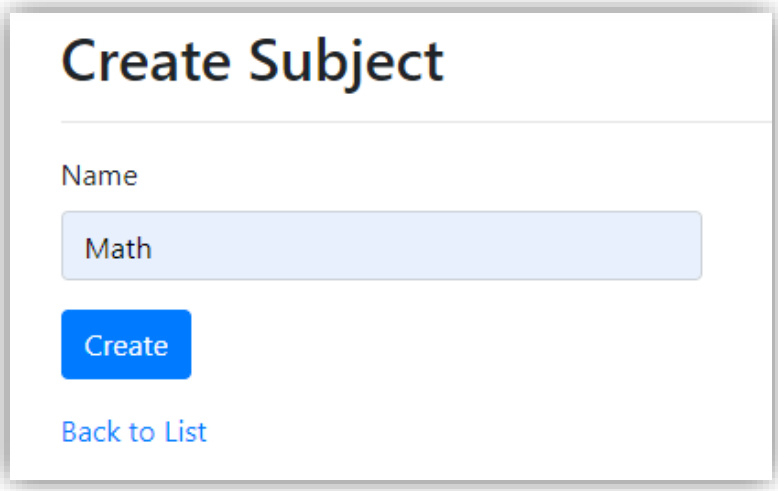

Рисунок 5.15 – Вікно створення предмету

Обравши опцію «Детальніше» користувач переходить до вікна деталей предмету (рисунок 5.16) та має можливість переглянути назву предмету та список тестів, які належать до цього предмету.

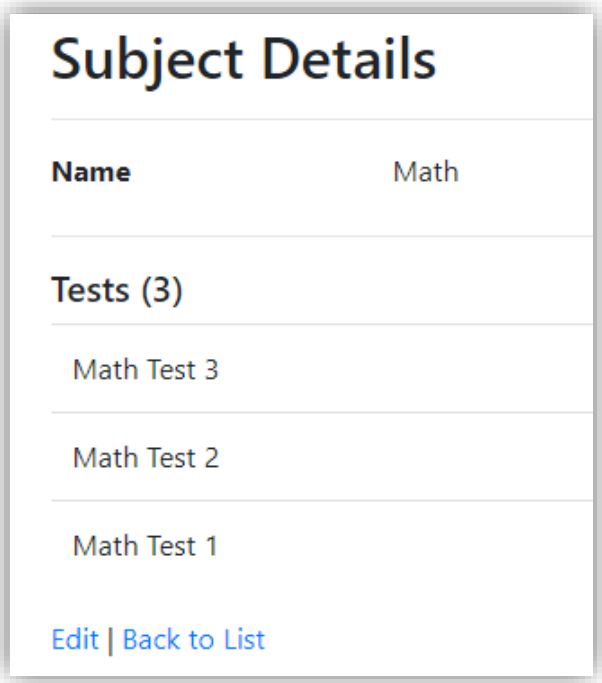

Рисунок 5.16 – Вікно деталей предмету

5.3.4 Розділ «Тести»

Перейшовши до розділу «Тести» користувачу відкривається вікно зі списком тестів (рисунок 5.17). Для кожного тесту відображається назва, тривалість, логічне значення, яке визначає чи потрібно перемішувати порядок питань під час проходження тесту, логічне значення, яке визначає чи потрібно перемішувати порядок варіантів відповідей у питаннях під час проходження тесту, предмет та доступні три опції: «Детальніше», «Редагування», «Видалення». Також користувач може додати новий тест застосувавши опцію «Створити новий».

Обравши опцію «Створити новий» або «Редагування» користувач переходить до вікна створення (рисунок 5.18) та редагування предмету відповідно. Вікна мають подібний вигляд. Користувач має можливість змінити назву, тривалість, логічне значення, яке визначає чи потрібно перемішувати порядок питань під час проходження тесту, логічне значення, яке визначає чи

потрібно перемішувати порядок варіантів відповідей у питаннях під час проходження тесту та предмет.

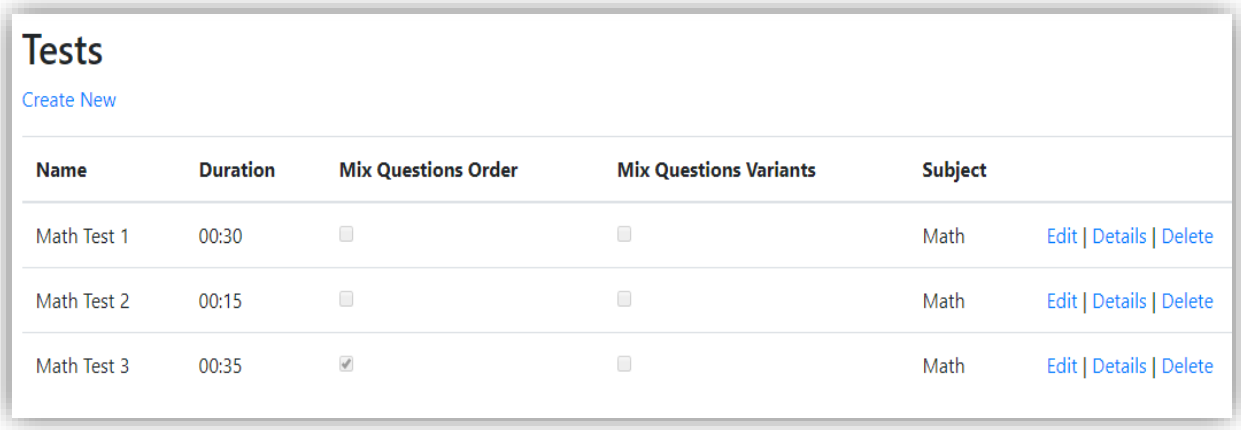

Рисунок 5.17 – Вікно зі списком тестів

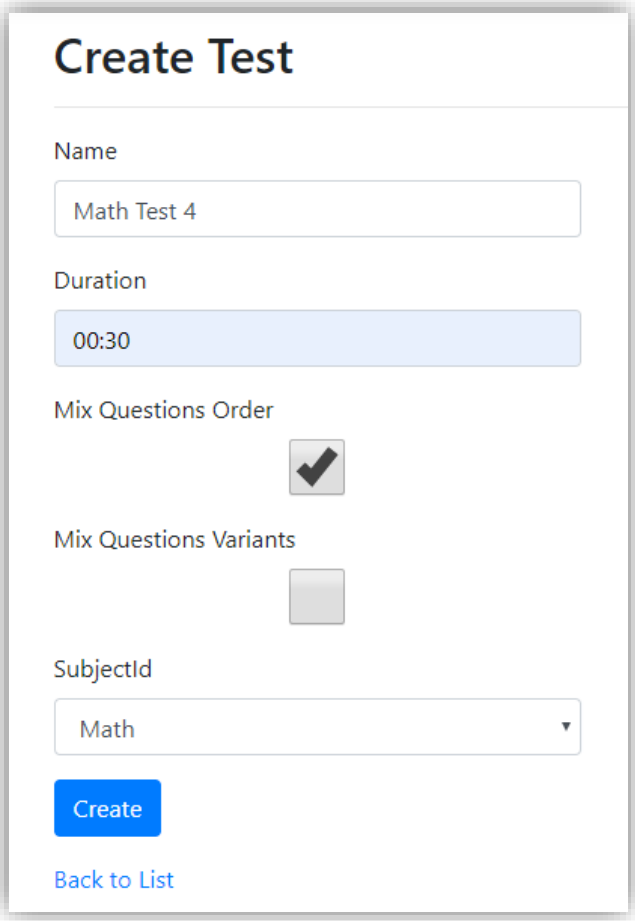

Рисунок 5.18 – Вікно створення тесту

Обравши опцію «Детальніше» користувач переходить до вікна деталей тесту (рисунок 5.19) та має можливість переглянути основну інформацію тесту, список питань тесту, а також список груп, студентам яких відкритий для проходження даний тест.

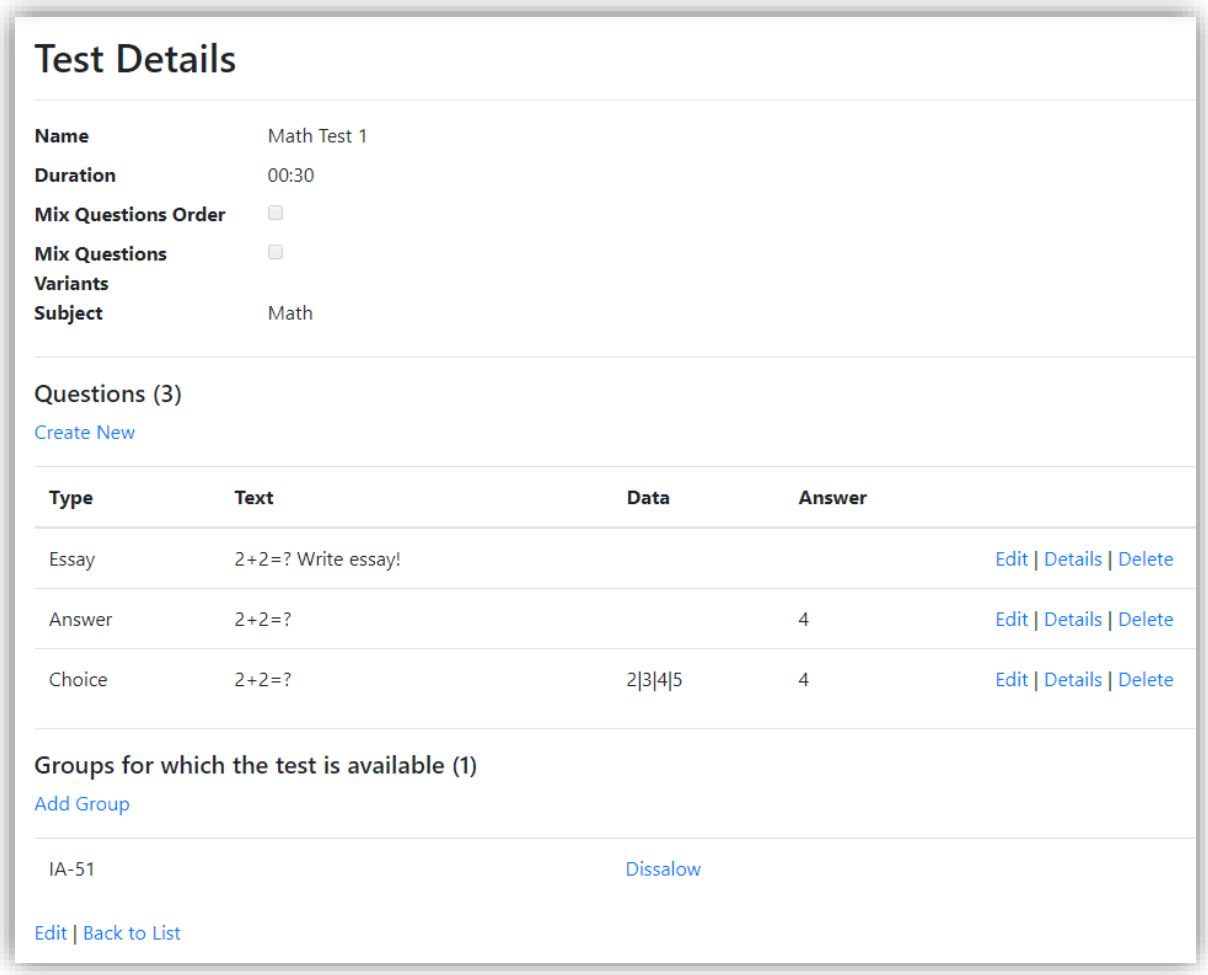

Рисунок 5.19 – Вікно деталей предмету

Для кожного питання тесту відображається тип питання, текст питання, данні питання, правильна відповідь на питання та доступні три опції: «Детальніше», «Редагування», «Видалення». Також користувач може додати нове питання застосувавши опцію «Створити нове». Обравши опцію «Створити нове» або «Редагування» користувач переходить до вікна створення та редагування питання (рисунок 5.20) відповідно. Вікна мають подібний вигляд.

Користувач має можливість змінити тип питання, текст питання, данні питання та правильну відповідь на питання.

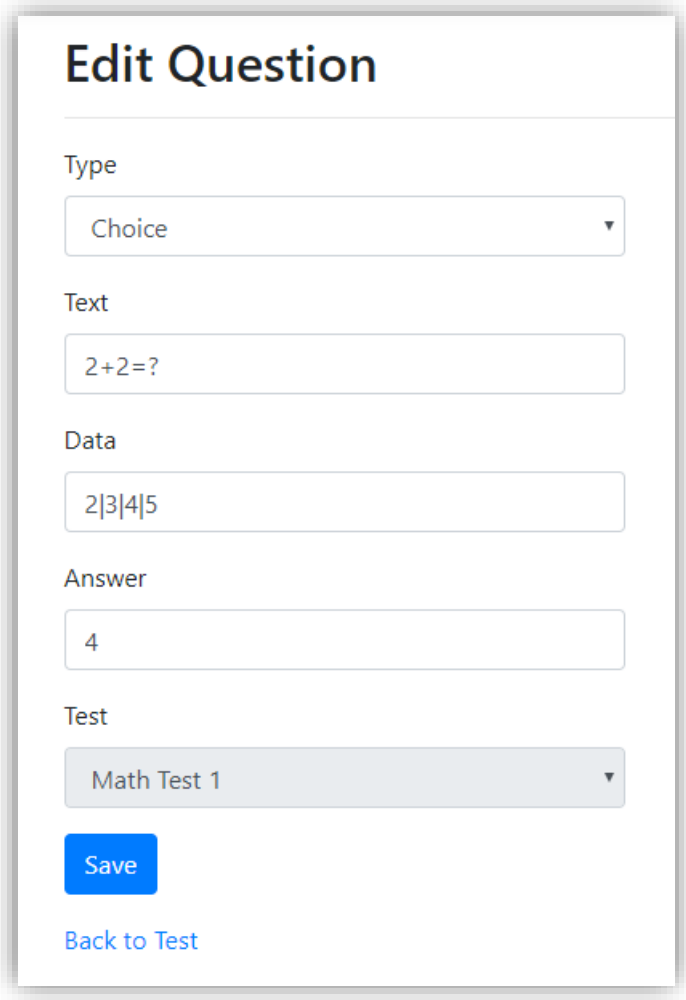

Рисунок 5.20 – Вікно редагування питання

Формат даних та правильної відповіді мають специфічний вигляд в залежності від типу питання, а саме:

1) Завдання закритої форми:
   a) Дані: «2|3|4|5», де 2, 3, 4, 5 – варіанти відповіді, а | – розділяючий варіанти символ;
   b) Відповідь: «4» або «4|5», де 4, 5 – варіанти відповіді, а | – розділяючий варіанти символ;

2) Завдання відкритої форми:
    a) Дані: відсутні;
    b) Відповідь: «текст відповіді»
3) Завдання на відповідність:
    a) Дані: «[2*2|3*3|4*4|5*5][4|9|16|25]», де 2*2, 3*3, 4*4, 5*5 – варіанти лівого стовпчика (що треба поставити у відповідність), 4, 9, 16, 25 – варіанти правого стовпчика (з чим треба поставити у відповідність), [] – розділяючі на варіанти лівого та правого стовпчика скобки, а | – розділяючий варіанти символ;
    b) Відповідь: «[2*2|4][3*3|9][4*4|16][5*5|25]», де [2*2|4], [3*3|9], [4*4|16], [5*5|25] – поставлені у відповідність варіанти відповіді з лівого та правого стовпчика, та розділені символом | ;
4) Завдання на встановлення правильної послідовності:
    a) Дані: «2|5|4|3», де 2, 3, 4, 5 – варіанти відповіді розташовані у неправильній послідовності, а | – розділяючий варіанти символ;
    b) Відповідь: «2|3|4|5»де 2, 3, 4, 5 – варіанти відповіді розташовані у правильній послідовності, а | – розділяючий варіанти символ;
5) Завдання відкритої форми по принципу есе:
    a) Дані: відсутні;
    b) Відповідь: відсутня

Обравши опцію «Додати групу» у секції списка груп, студентам яких відкритий для проходження даний тест користувач переходить до вікна надання доступу групі до тесту (рисунок 5.21) де він має можливість обрати групу.

Також користувач може заборонити групі доступ до тесту, скориставшись опцією «Заборонити» навпроти необхідної групи.

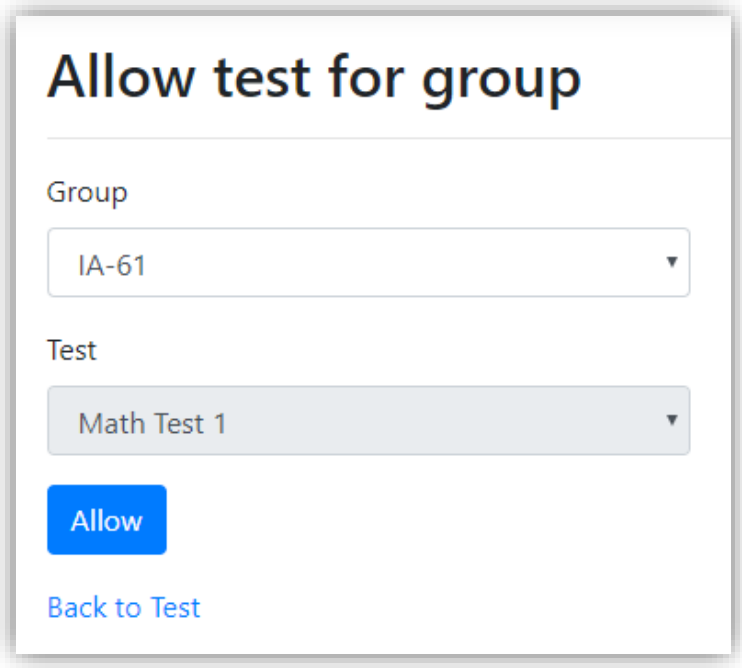

Рисунок 5.21 – Вікно надання доступу групі до тесту

5.3.5 Розділ «Тестові сесії»

Перейшовши до розділу «Тестові сесії» користувачу відкривається вікно зі списком тестових сесій (рисунок 5.22). Для кожної тестової сесії відображається дата початку, дата завершення, статус, час, який залишився на проходження тесту, назва тесту, ім’я студента та доступна одна опція: «Детальніше».

Обравши опцію «Детальніше» користувач переходить до вікна деталей тестової сесії (рисунок 5.23) та має можливість переглянути основну інформацію, та інформацію про відповіді на питання тесту.

У секції відповідей на питання тесту відображається кількість питань в тесті, кількість правильних відповідей та кількість відповідей, що чекають на оцінювання викладачем. Для кожної відповіді відображається тип питання, текст питання, данні питання, відповідь студента, логічне значення, яке зазначає

чи правильна відповідь та доступна одна з опцій: «Позначити як правильна» або «Позначити як неправильна» в залежності від того, яким чином вже позначена відповідь. Для всіх питань, окрім питань відкритої форми по принципу есе система самостійно визначаю правильність відповіді.

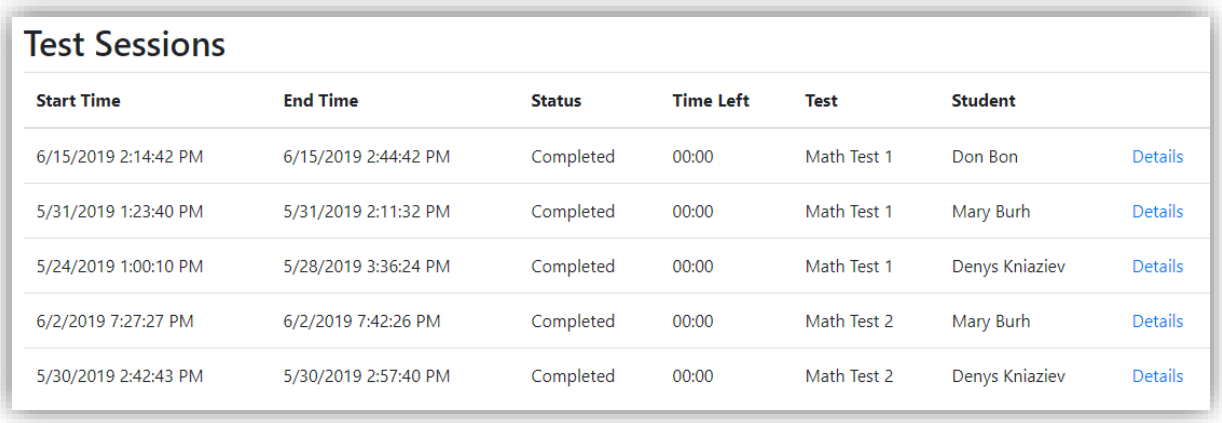

Рисунок 5.22 – Вікно зі списком тестових сесій

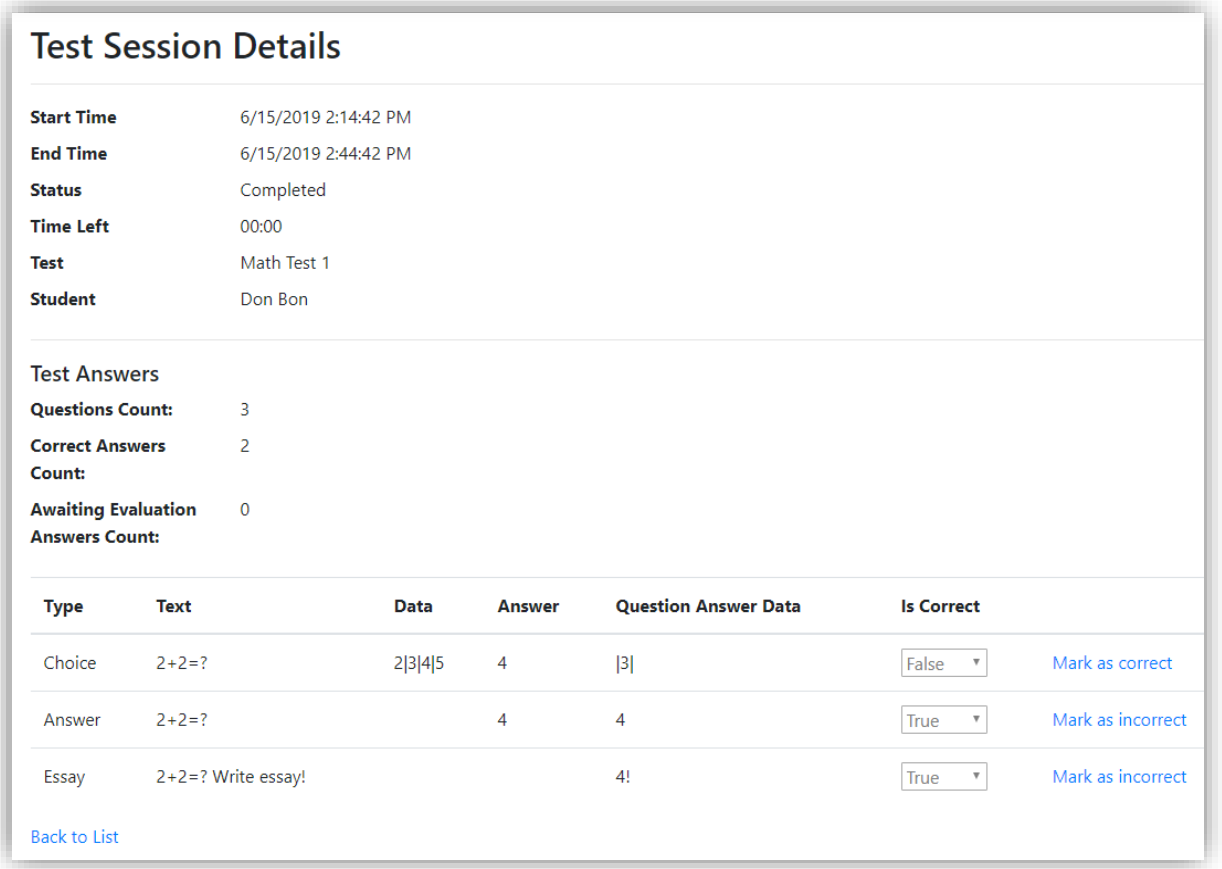

Рисунок 5.23 – Вікно деталей тестової сесії

5.4 Робочій простір студента

Користувач системи з правами студента має доступ тільки до розділу системи «Тести». Перейшовши до розділу «Тести» користувачу відкривається вікно зі списком тестів (рисунок 5.24). Для кожного тесту відображається назва, тривалість, предмет та доступна одна з опцій: «Розпочати» або «Результат».

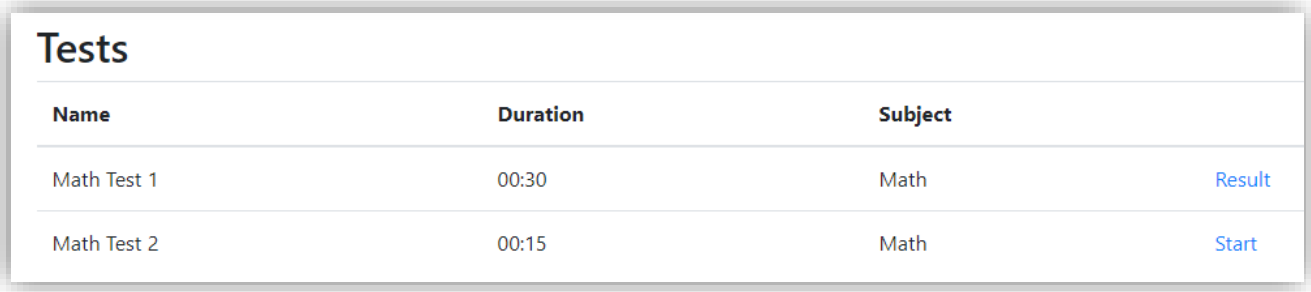

Рисунок 5.24 – Вікно зі списком тестів

Якщо проходження тесту ще не розпочато студенту доступна опція «Розпочати» тест. Обравши опцію «Розпочати» користувач переходить до вікна підтвердження початку тестової сесії (рисунок 5.25) та має можливість переглянути основну інформацію про тест і розпочати його.

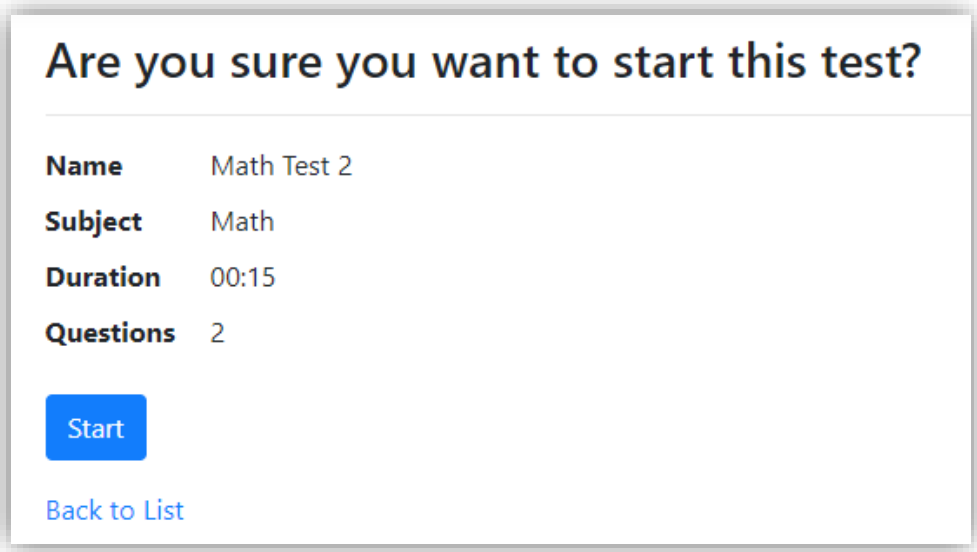

Рисунок 5.25 – Вікно підтвердження початку тестової сесії

Після підтвердження початку тесту користувач переходить до вікна тестової сесії (рисунок 5.26).

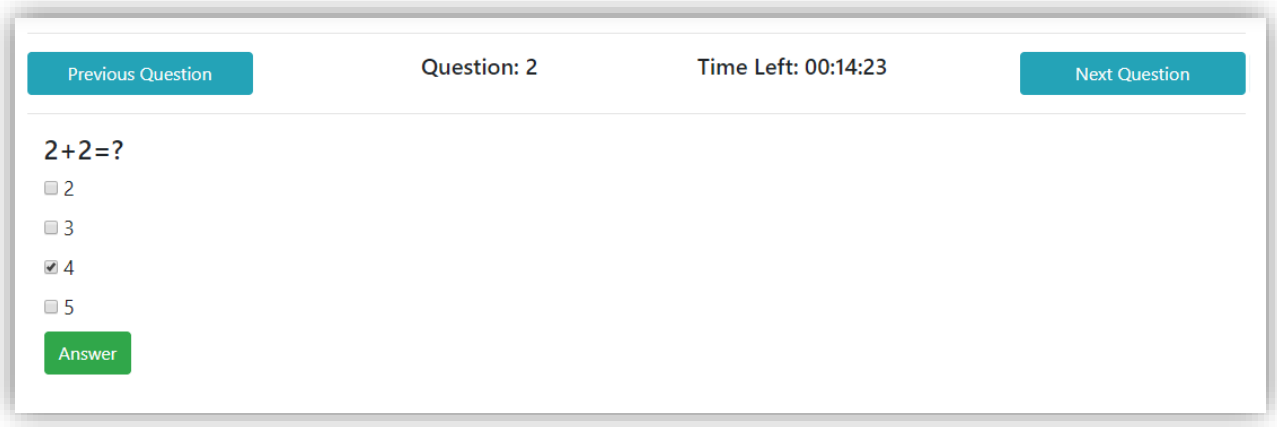

Рисунок 5.26 – Вікно тестової сесії

Студент має можливість переглядати питання, відповідати на них, а також переходити до попереднього та наступного питання. При переході між питаннями система зберігає підтверджені відповіді. У верхній частині вікна користувачу відображається таймер, який показує кількість часу, який залишився на проходження тесту.

Після завершення часу тестової сесії користувач перенаправляється до вікна результатів тесту (рисунок 5.27), де відображається кількість питань в тесті, кількість правильних відповідей та кількість відповідей, що чекають на оцінювання викладачем.

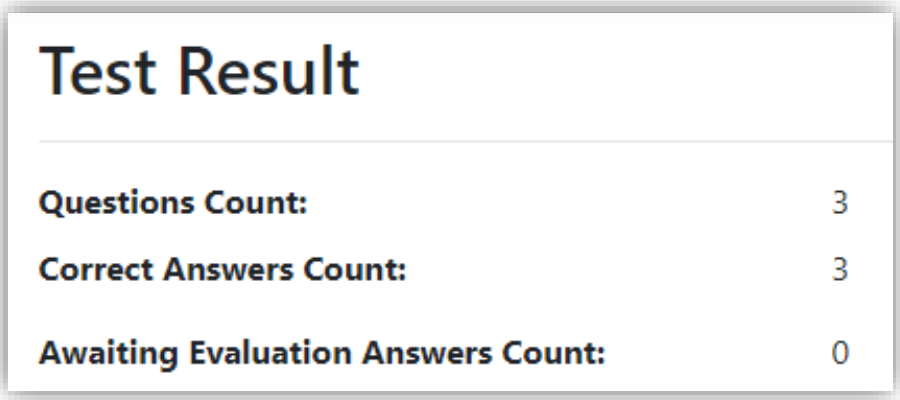

Рисунок 5.27 – Вікно тестової сесії

ВИСНОВКИ

В ході виконання дипломної роботи було здійснено проектування архітектури програмного продукту і бази даних, зроблено вибір програмних засобів і технологій для реалізації проекту, реалізована система і база даних, розроблена програмна документація для супроводу і системно-технічної підтримки продукту.

При розробці системи були проаналізовані сучасні web-технології, що дозволяють створювати інтерактивні web-орієнтовані програмні продукти. Для розробки системи в рамках поставленого завдання було обрано наступні технології: ASP.NET Core, Entity Framework Core, ASP.Net Identity, PostgreSQL, JavaScript, HTML, CSS. Програмний продукт було розроблено за допомогою IDE Visual Studio 2017.

В ході роботи було проаналізовано декілька основних систем аналогів, виявлена багата кількість плюсів і мінусів, які були враховані при розробці даного програмного забезпечення. Створений програмний продукт відповідає всім вимогам, висунуті на етапі постановки завдання. Багатоплатформенність стеку технологій дозволить розгортати систему на платформах Windows, MacOS та Linux.

ПЕРЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Теория и практика дистанционного обучения: Учеб. пособие для студ. высш. пед. учебн. заведений / Е. С. Полат, М. Ю. Бухаркина, М. В. Моисеева; Под ред. Е. С. Полат // М.: Издательский центр «Академия», 2004. — 416 с.
2. Сайт програми PikaTest [Електронний ресурс] – Режим доступу до ресурсу: http://kripexx.narod.ru/pikatest/.
3. Сайт програми UniTest [Електронний ресурс] – Режим доступу до ресурсу: http://unitest.sfu-kras.ru/.
4. Сайт програми INDIGO [Електронний ресурс] – Режим доступу до ресурсу: http://www.indigotech.ru/.
5. Работа в системе дистантного обучения Moodle. Инструкция для преподавателей ./ В.В. Гвоздев, В.В. Проскурин/ Тольятти, 2011. – 155 с.
6. The ASP.Net Site [Електронний ресурс] – Режим доступу до ресурсу: http://www.asp.net/.
7. Entity Framework [Електронний ресурс] – Режим доступу до ресурсу: http://www.asp.net/entity-framework.
8. JQuery Site [Електронний ресурс] – Режим доступу до ресурсу: https://jquery.com/.
9. .NET Core and Open-Source [Електронний ресурс] – Режим доступу до ресурсу: https://msdn.microsoft.com/en-us/library/dn878908 (v=vs.110).aspx.
10. Справочник «Паттерны проектирования» [Електронний ресурс] – Режим доступу до ресурсу: http://design-pattern.ru/patterns/mvc.html.
11. ADO.NET Blog [Електронний ресурс] – Режим доступу до ресурсу: http://blogs.msdn.com/b/adonet/.
12. Introduction to ASP.NET Identity [Електронний ресурс] – Режим доступу до ресурсу: http://www.asp.net/identity/overview/getting-started/introduction-to-aspnet-identity.

13. Inversion of Control Containers and the Dependency Injection pattern [Електронний ресурс] – Режим доступу до ресурсу: http://martinfowler.com/articles/injection.html.

14. What is Code-First? [Електронний ресурс] – Режим доступу до ресурсу: http://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx.

15. Code First Migrations [Електронний ресурс] – Режим доступу до ресурсу: https://msdn.microsoft.com/en-us/data/jj591621.aspx.

16. Create, Retrieve, Update and Delete (CRUD) [Електронний ресурс] – Режим доступу до ресурсу: https://www.techopedia.com /definition/25949/create-retrieve-update-and-delete-crud.

17. Поняття архітектури програмного забезпечення. [Електронний ресурс]. — Режим доступу до ресурсу: http://www.lib.mdpu.org.ua/e-book/web/Lec10.html

18. Сравнение многоуровневых и клиент-серверных систем [Електронний ресурс]. — Режим доступу до ресурсу: http://msdn.microsoft.com/ru-ru/library/ee895340.aspx

Додаток А

Лістинг компонентів додатку
(обов'язковий)

Лістинг А.1 – Клас Test

```
public class Test : Entity<int>
{
    [Required]
    [StringLength(100, ErrorMessage = "The {0} must be at max {1} characters
long.")]
    public string Name { get; set; }

    [Required]
    [DisplayFormat(DataFormatString = "{0:hh\\:mm}", ApplyFormatInEditMode = true)]
    public TimeSpan Duration { get; set; }

    [Required]
    [Display(Name = "Mix Questions Order")]
    public bool MixQuestionsOrder { get; set; }

    [Required]
    [Display(Name = "Mix Questions Variants")]
    public bool MixQuestionsVariants { get; set; }

    [Required]
    public int SubjectId { get; set; }
    public Subject Subject { get; set; }

    public List<Question> Questions { get; set; } = new List<Question>();

    [Display(Name = "Test Sessions")]
    public List<TestSession> TestSessions { get; set; } = new List<TestSession>();

    [Display(Name = "Group Tests")]
    public List<GroupTest> GroupTests { get; set; } = new List<GroupTest>();
}
```

Лістинг А.2 – Клас TestsController

```
[Area("Admin")]
[Authorize(Roles = "Admin")]
public class TestsController : Controller
{
    private readonly IApplicationDbContext _context;

    public TestsController(IApplicationDbContext context)
    {
        _context = context;
    }

    // GET: Tests
    public async Task<IActionResult> Index()
    {
        var applicationDbContext = _context.Tests.Include(t => t.Subject);
        return View(await applicationDbContext.ToListAsync());
    }
```

```
        // GET: Tests/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var test = await _context.Tests
                .Include(t => t.Subject)
                .Include(t => t.Questions)
                .Include(t => t.GroupTests)
                    .ThenInclude(t => t.Group)
                .FirstOrDefaultAsync(m => m.Id == id);

            if (test == null)
            {
                return NotFound();
            }

            return View(test);
        }

        // GET: Tests/Create
        public IActionResult Create()
        {
            ViewData["SubjectId"] = new SelectList(_context.Subjects, "Id", "Name");
            return View();
        }

        // POST: Tests/Create
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult>
Create([Bind("Name,Duration,MixQuestionsOrder,MixQuestionsVariants,SubjectId,Id")] Test
test)
        {
            if (ModelState.IsValid)
            {
                _context.Tests.Add(test);
                await _context.CommitAsync();
                return RedirectToAction(nameof(Index));
            }
            ViewData["SubjectId"] = new SelectList(_context.Subjects, "Id", "Name",
test.SubjectId);
            return View(test);
        }

        // GET: Tests/Edit/5
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var test = await _context.Tests.FindAsync(id);
            if (test == null)
            {
                return NotFound();
            }
            ViewData["SubjectId"] = new SelectList(_context.Subjects, "Id", "Name",
test.SubjectId);
```

```
            return View(test);
        }

        // POST: Tests/Edit/5
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id,
[Bind("Name,Duration,MixQuestionsOrder,MixQuestionsVariants,SubjectId,Id")] Test test)
        {
            if (id != test.Id)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Tests.Update(test);
                    await _context.CommitAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!TestExists(test.Id))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
                return RedirectToAction(nameof(Index));
            }
            ViewData["SubjectId"] = new SelectList(_context.Subjects, "Id", "Name",
test.SubjectId);
            return View(test);
        }

        // GET: Tests/Delete/5
        public async Task<IActionResult> Delete(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var test = await _context.Tests
                .Include(t => t.Subject)
                .FirstOrDefaultAsync(m => m.Id == id);
            if (test == null)
            {
                return NotFound();
            }

            return View(test);
        }

        // POST: Tests/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
```

```
            var test = await _context.Tests.FindAsync(id);
            _context.Tests.Remove(test);
            await _context.CommitAsync();
            return RedirectToAction(nameof(Index));
        }

        private bool TestExists(int id)
        {
            return _context.Tests.Any(e => e.Id == id);
        }
    }
```

Лістинг А.3 – Клас TestSessionsActualizationService

```
    public class TestSessionsActualizationService : IHostedService
    {
        private const int TimerIntervalSeconds = 1;

        private Timer _timer;
        private readonly IServiceScopeFactory _factory;

        public TestSessionsActualizationService(IServiceScopeFactory factory)
        {
            _factory = factory;
        }

        public Task StartAsync(CancellationToken cancellationToken)
        {
            _timer = new Timer(async state =>
            {
                using (var scope = _factory.CreateScope())
                {
                    var context =
scope.ServiceProvider.GetRequiredService<IApplicationDbContext>();
                    var testSessions = context.TestSessions.Where(x => x.Status ==
TestSessionStatus.InProgress);
                    foreach (var testSession in testSessions)
                    {
                        var newTimeLeftSeconds = testSession.TimeLeft.TotalSeconds -
TimerIntervalSeconds;
                        if (newTimeLeftSeconds <= 0)
                        {
                            testSession.Status = TestSessionStatus.Completed;
                            testSession.TimeLeft = TimeSpan.Zero;
                            testSession.EndTime =
scope.ServiceProvider.GetRequiredService<IDateTimeProvider>().UtcNow;
                        }
                        else
                        {
                            testSession.TimeLeft =
TimeSpan.FromSeconds(newTimeLeftSeconds);
                        }
                        context.TestSessions.Update(testSession);
                    }
                    await context.CommitAsync();
                }
            }, null, TimerIntervalSeconds * 1000, TimerIntervalSeconds * 1000);

            return Task.CompletedTask;
        }

        public Task StopAsync(CancellationToken cancellationToken)
```

```
        {
            _timer?.Dispose();
            return Task.CompletedTask;
        }
    }
```

Лістинг А.4 – Клас ApplicationDbContext

```
    public class ApplicationDbContext : DbContext, IApplicationDbContext
    {
        private readonly IDateTimeProvider _dateTimeProvider;

        public DbSet<Group> Groups { get; set; }
        public DbSet<GroupTest> GroupTests { get; set; }
        public DbSet<Student> Students { get; set; }
        public DbSet<Subject> Subjects { get; set; }
        public DbSet<Test> Tests { get; set; }
        public DbSet<TestSession> TestSessions { get; set; }
        public DbSet<Question> Questions { get; set; }
        public DbSet<QuestionAnswer> QuestionAnswers { get; set; }

        public async Task CommitAsync()
        {
            AddTrackingInfo();
            await SaveChangesAsync();
        }

        public ApplicationDbContext(
            DbContextOptions options,
            IDateTimeProvider dateTimeProvider)
            : base(options)
        {
            _dateTimeProvider = dateTimeProvider;
        }

        private void AddTrackingInfo()
        {
            var entries = ChangeTracker.Entries().Where(x =>
                x.Entity is ITrackable && (x.State == EntityState.Added || x.State ==
EntityState.Modified));

            foreach (var entry in entries)
            {
                var entity = (ITrackable)entry.Entity;

                if (entry.State == EntityState.Added)
                    entity.CreatedOn = _dateTimeProvider.UtcNow;

                entry.Property(nameof(ITrackable.CreatedOn)).IsModified = false;
                entity.ModifiedOn = _dateTimeProvider.UtcNow;
            }
        }
    }
```

Лістинг А.5 – Клас AdminTokenAttribute

```
    public class AdminTokenAttribute : ValidationAttribute
    {
```

```
        protected override ValidationResult IsValid(object value, ValidationContext
validationContext)
        {
            var adminTokenPassed = (string) value;
            var adminTokenExpected =
validationContext.GetRequiredService<IOptions<AdminOptions>>().Value.AdminToken;

            return adminTokenPassed == adminTokenExpected
                ? ValidationResult.Success
                : new ValidationResult("Invalid Admin token");
        }
    }
```

Лістинг А.6 – Клас RegisterAdminModel

```
    [AllowAnonymous]
    public class RegisterAdminModel : PageModel
    {
        private readonly SignInManager<IdentityUser> _signInManager;
        private readonly UserManager<IdentityUser> _userManager;
        private readonly ILogger<RegisterModel> _logger;
        private readonly IEmailSender _emailSender;

        public RegisterAdminModel(
            UserManager<IdentityUser> userManager,
            SignInManager<IdentityUser> signInManager,
            ILogger<RegisterModel> logger,
            IEmailSender emailSender)
        {
            _userManager = userManager;
            _signInManager = signInManager;
            _logger = logger;
            _emailSender = emailSender;
        }

        [BindProperty]
        public InputModel Input { get; set; }

        public string ReturnUrl { get; set; }

        public class InputModel
        {
            [Required]
            [EmailAddress]
            [Display(Name = "Email")]
            public string Email { get; set; }

            [Required]
            [StringLength(100, ErrorMessage = "The {0} must be at least {2} and at max
{1} characters long.", MinimumLength = 6)]
            [DataType(DataType.Password)]
            [Display(Name = "Password")]
            public string Password { get; set; }

            [DataType(DataType.Password)]
            [Display(Name = "Confirm password")]
            [Compare("Password", ErrorMessage = "The password and confirmation password
do not match.")]
            public string ConfirmPassword { get; set; }

            [Required]
            [AdminToken]
            [DataType(DataType.Password)]
```

```
            [Display(Name = "Admin token")]
            public string AdminToken { get; set; }
        }

        public void OnGet(string returnUrl = null)
        {
            ReturnUrl = returnUrl;
        }

        public async Task<IActionResult> OnPostAsync(string returnUrl = null)
        {
            returnUrl = returnUrl ?? Url.Content("~/");

            if (ModelState.IsValid)
            {
                var user = new IdentityUser { UserName = Input.Email, Email =
Input.Email };

                var createResult = await _userManager.CreateAsync(user,
Input.Password);
                var addToRoleResult = await _userManager.AddToRoleAsync(user, "Admin");

                if (createResult.Succeeded && addToRoleResult.Succeeded)
                {
                    _logger.LogInformation("User created a new account with
password.");

                    var code = await
_userManager.GenerateEmailConfirmationTokenAsync(user);
                    var callbackUrl = Url.Page(
                        "/Account/ConfirmEmail",
                        pageHandler: null,
                        values: new { userId = user.Id, code },
                        protocol: Request.Scheme);

                    await _emailSender.SendEmailAsync(Input.Email, "Confirm your
email",
                        $"Please confirm your account by <a
href='{HtmlEncoder.Default.Encode(callbackUrl)}'>clicking here</a>.");

                    await _signInManager.SignInAsync(user, isPersistent: false);
                    return LocalRedirect(returnUrl);
                }
                foreach (var error in createResult.Errors)
                {
                    ModelState.AddModelError(string.Empty, error.Description);
                }
                foreach (var error in addToRoleResult.Errors)
                {
                    ModelState.AddModelError(string.Empty, error.Description);
                }
            }

            return Page();
        }
    }
```